A Prototype Toolset for the ArcGIS Hydro Data Model

by

Timothy Lee Whiteaker, B.S.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

August 2001

A Prototype Toolset for the ArcGIS Hydro Data Model

Approved by
Supervising Committee:

David R. Maidment

Howard M. Liljestrand

Francisco Olivera

Acknowledgments

I would like to thank Dr. David Maidment of the Center for Research in Water Resources for his guidance and enthusiasm. I would also like to thank Camp Dresser & McKee Inc. for their generous support of my research. Thanks go to David Arctur and Steve Grise of Environmental Systems Research Institute Inc. for providing valuable assistance and insights. Thanks also go to Dr. Francisco Olivera for his enlightening perspective. I would like to give a special thanks to the CRWR family (Mom, Dad, sisters, pals) for making this journey a blast. Finally, I would like to thank Dr. Vincent Neary of Tennessee Technological University for encouraging me to further my knowledge of hydrology and GIS at the graduate level.

August 17, 2001

Abstract

A Prototype Toolset for the ArcGIS Hydro Data Model

Timothy Lee Whiteaker, M.S.E.

The University of Texas at Austin, 2001

Supervisor: David R. Maidment

The incorporation of a COM-compliant design, object-oriented programming, and relational database technology into Geographic Information Systems has opened the door for the next generation of GIS-enabled hydrologic simulation models to be developed. The ArcGIS Hydro Data Model (Arc Hydro) helps to bridge the gap between GIS and computational models by facilitating the preparation of GIS data for model use. This thesis describes some approaches for developing a simulation model on top of Arc Hydro, discusses object-oriented programming concepts, and provides software construction guidelines. The result is a set of tools which operate on Arc Hydro in order to prepare GIS data for use in a simulation model.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Through recent innovations in Geographic Information Systems technology, users can now create custom objects with special properties and behaviors to represent geographic features. The ArcGIS Hydro Data Model uses this technology to store water resources features in a manner which facilitates the conversion of the data for use in a water resources simulation model. This thesis explores methods that may be used to build a simulation model on top of the ArcGIS Hydro Data Model, and presents a prototype toolset which operates on the ArcGIS Hydro Data Model.

1.1 BACKGROUND

A Geographic Information System (GIS) is a computer system that enables the creation, manipulation, analysis, storing, and display of geographically referenced information. By incorporating database technology and a suite of both proprietary and non-proprietary tools, GIS allows users to examine spatial relationships that might otherwise have been overlooked in the decision-making process. GIS was used primarily for display purposes during its early stages, with applications in the transportation plans of Detroit and Chicago (Goodchild and Kemp, 1990). In the 1980s, advancements in both hardware and software technology enabled GIS to become available on platforms affordable by a much larger group of potential users (Goodchild and Kemp, 1990), while also extending the analysis capabilities of the software. Today, GIS technology can be found in thousands of organizations around the world, including municipalities,

crime investigation units, forestry services, and Internet-based mapping services such as MapQuest.

In 2000, GIS entered a new era in technological advancement when the Environmental Systems Research Institute (ESRI), the world's leading producer of GIS software, released the latest versions of its ArcInfo and ArcView GIS software. Collectively known as ArcGIS, this software system enables a truly object-oriented representation of geographic features. By creating COM-standard interfaces, users can associate properties and behaviors with features (COM is defined in section 2.2.2.). This means that in addition to appearing as a simple blue line on the map, the GIS representation of a river can possess attributes that more accurately model its real-world counterpart. Because ArcGIS is COM-compliant, it can easily communicate with other COM-compliant software, such as Microsoft Word or Internet Explorer. ArcGIS can also be customized using Visual Basic and can utilize all other COM-compliant object libraries registered on the machine where the software is installed. ArcGIS also employs innovations in database technology that allow tables to possess a blob field, which can store anything from an integer to an image to a file. By storing the spatial representation of features in a blob field, ArcGIS more tightly couples the spatial data with its attribute information.

To take advantage of ArcGIS's new and innovative capabilities, a consortium for GIS in water resources coordinated by the Center for Research in Water Resources at the University of Texas at Austin has developed a data model to represent water resources features, in both a hydrologic and hydrographic

sense, by creating a series of custom objects with their own attributes and behaviors. The ArcGIS Hydro data model (Arc Hydro), as the model is called, is not meant to serve as a sophisticated hydrologic simulation model. As a data model, its primary purpose is to provide a geospatial framework for storing water resources data, with some simple methods that facilitate data development for use in a simulation model, such as HEC-HMS or MIKE-SHE. Arc Hydro attempts to integrate the extensive resources of cartographic information available with the components required as input to water resources computational models by providing a structure that accommodates both types of data as well as conversion and some overlap between data types.

Since its conception two years ago, ArcGIS Hydro data model has undergone numerous reviews and is now completed, at least structurally speaking. The next step in Arc Hydro's development is to examine to what extent it supports the linkage between GIS and a computational model. The purpose of this research is to explore the methods available, given the new innovations in technology, to create a water resources simulation model on top of the ArcGIS Hydro Data Model.

1.2 MOTIVATION

Hydrologic simulation models have existed for decades. The routines to perform hydrologic computations are so well understood, and the models themselves have been validated through so many years of use, that few models have undergone significant changes in recent years. However, with the latest advances in GIS technology, there are new benefits to be gained by linking GIS

with hydrologic models. ArcGIS geodatabases use the familiar and accessible structure of Relational Database Management Systems (RDBMS) to store GIS data, making ArcGIS the first GIS software produced by ESRI that does not rely on a proprietary file format to store the spatial representation of features. The COM-compliant nature of ArcGIS also improves compatibility with other software, meaning that ArcGIS is able to communicate with hydrologic simulation models in a manner that was previously not possible.

In addition to bringing GIS and hydrologic software closer together, the potential also exists to narrow the gap between GIS data and inputs to a hydrologic model. Traditionally, a hydrologic model can use only its own type of data format for model inputs. While data can be created in a model to represent components of reality, typically a great storehouse of data already exists in the form of a GIS. Incorporating this data into a hydrologic model can save a great deal of time and avoid the "redundant" creation of data. GIS utilities such as CRWR Pre-Pro can already export GIS data for use in hydrologic models (in this case HEC-HMS,) but these utilities typically write the data to a text file, which is later imported into the model (Olivera, 2001). With COM-compliance, this data conversion could happen in a single step if the object libraries of the GIS and the model are compatible. In fact, it may be possible to use a single source of data for both the GIS and the model. Thus, when a parameter is added or changed on one end, that change is reflected in the other software as well. The ability of ArcGIS to support custom objects makes this connection even more attainable. Instead of a computational model being forced to use a standard GIS data format, custom

objects could be developed that possess the properties and methods necessary to function in both the GIS and the model.

ArcGIS allows the user to create specially defined classes known as custom feature classes that extend the power and functionality of GIS features. The ArcGIS Hydro data model is the first attempt at using custom feature classes in ArcGIS to improve the connection between GIS data and the requirements for a computational hydrologic or hydraulic model. Arc Hydro contains custom feature classes to represent cartographic features, drainage patterns, channel systems, and the movement of water through the landscape. Yet while the structural representation of these features is complete, no methods have been attached to any feature classes in Arc Hydro. In addition, Arc Hydro has not yet been used to prepare data for a water resources computational model, so the effectiveness with which it supports the link between GIS and hydrologic models is not known. Exploring different strategies to developing a computational model on top of the ArcGIS Hydro data model will not only test its ability to support this link, but it may also provide insight as to the appropriate methods that should be attached to the custom objects in Arc Hydro. This research will also lead to a better understanding of how GIS and a computational hydrologic model can be connected in general.

1.3 OBJECTIVE AND SCOPE

The purpose of this research is to explore methods to create a computational hydrologic or hydraulic model on top of the ArcGIS Hydro data model. As such, the procedures outlined and the resulting tools developed are

intended to be applied within the framework of the ArcGIS software and the ArcGIS Hydro data model. However, some of the discussion involves general principles behind the application of object-oriented concepts towards hydrologic and hydraulic modeling, and may provide value outside of the ArcGIS context.

The goal of this thesis is not to create a computational model, but to determine the various methods that may be used to create such a model given the latest innovations in software technology and the evolution of the ArcGIS Hydro data model. Advances in object-oriented programming and RDBMS technology have dramatically increased the functionality, compatibility, and power of GIS. These advances may have opened up new avenues for integrating computational models with geographic data. This thesis attempts to discover those avenues and learn how they can be applied from within the context of the ArcGIS Hydro Data Model.

Another goal of this research is to examine the effectiveness with which Arc Hydro links GIS data and computational models. Currently, no classes in Arc Hydro possess behavior. By exploring different needs of hydrologic computational models, potential behaviors (or methods of preparing data) for Arc Hydro classes can be formulated.

1.4 THESIS OUTLINE

This thesis is divided into six chapters. The first provides some background information about Geographic Information Systems and the ArcGIS Hydro data model, followed by an overview of the motivation, scope, and objective in developing a computational model on top of Arc Hydro. The second

provides some general concepts about object-oriented modeling and reviews the principles of sound software construction and user interface design. This chapter also describes other object-oriented hydrologic and hydraulic models that were studied to gain a better understanding of model development. The third chapter provides an overview of some innovations behind the design of the ArcGIS software, and introduces the components of the ArcGIS Hydro data model. The application of object-oriented principles towards hydrologic model design is also discussed in this chapter. This chapter concludes with a description of development approaches and interface design techniques for computational model development in relation to the ArcGIS Hydro data model. The fourth chapter details the steps taken to create a set of custom tools that operate on Arc Hydro, followed by a discussion of how software design principles were applied to the development of the tools. The fifth describes each tool that was developed, with a discussion of benefits and limitations for each tool. The sixth draws conclusions and suggests future work that may be undertaken.

Chapter 2: Literature Review

Before the first line of code is written for a computational model, a good understanding of proper software construction techniques should be acquired. Meyer (1997) outlines some key concepts related to intelligent software design in Object-Oriented Software Construction. These concepts lend themselves to object-oriented programming, which has been the programming style of choice since the early 1990s. Aside from the structure of a software's design, perhaps an equally important component of a software system is the interface through which communication occurs between the user and the program. Hartley (1998) provides some instruction on graphical user interface design. All of these concepts can be used to evaluate object-oriented models in the water resources realm that have already been created. Indeed, a study of existing models has proven useful in determining effective approaches in developing a computational model on top of the ArcGIS Hydro data model.

2.1 Software Design Principles

Meyer (1997) provides some widely accepted guidelines on intelligent software construction. The key indicator of a program's quality is the degree to which it successfully performs the tasks outlined in the solution to the problem statement. However, in addition to functioning correctly, a software product should also be *robust*, *extensible*, *reusable*, and *compatible* (Meyer, 1997).

2.1.1 Robustness

Robustness refers to the ability of software to function when conditions fall outside of the specification made by the problem statement. No matter how meticulously the designer plans the software development, situations will always occur outside the designer's specifications. A robust software system will handle these unexpected situations gracefully, without crashing or producing other catastrophic events (Meyer, 1997).

2.1.2 Extensibility

Extensibility refers to the ease with which adaptations may be made to software in order to meet changing needs or specifications. Typically, the larger the software system, the more difficult it is to make changes due to a more and more complex interconnection between software components. By following two principles, design simplicity and decentralization, software extensibility can be dramatically improved. The concept behind design simplicity is that simple designs are easier to change than complex ones. If the designer is not careful, a software system can become so convoluted that making changes to the system design requires more time than simply rebuilding the system from scratch. The principle of decentralization states that as software modules become more autonomous, changes in software design will be more likely to affect a single module, rather than requiring changes throughout a large chain of modules (Meyer, 1997).

2.1.3 Reusability

Reusability measures the ability of a software system's components (or the system itself) to be reused for different applications. Often different applications will require similar tasks to be performed, such as closing a window or opening a file. Being able to reuse common components between software applications saves much time in the development process. Reusing software components also provides a quality check in many situations, as those components have already been tested, debugged, and proven through previous applications (Meyer, 1997).

2.1.4 Compatibility

Compatibility measures the ability of a software system to be combined with others. Compatibility is becoming more and more important as developers seek complete and integrated solutions that follow a problem from its initial statement to its final, presentable solution output. A software product that operates in an isolated environment will soon find itself inadequate and obsolete, while a product that can work with Internet, word processing, database, multimedia, or other utilities will remain useful components of a total solution (Meyer, 1997).

2.2 OBJECT-ORIENTED PROGRAMMING

Many of the design principles stated above describe the very nature of Object-oriented programming. When combined with COM, an object-oriented approach can lead to a solution that can be integrated with many other software systems.

2.2.1 Concepts

Object-oriented programming uses fundamental constructs called objects to represent real-world concepts. Objects possess both a data structure and behavior. An object's data structure is described by its properties or attributes. A *property* is a descriptor for an object that may take on different values. For example, a river object could be described by a width property. An object's behavior is also known as its methods or operations. A *method* is a task that an object performs. For instance, a river object might have a method that routes a hydrograph through it. Object-oriented development is a thought process and is largely independent of its actual implementation in a programming language. By focusing first on the design of objects rather than implementation, designers can create objects that best model the relevant aspects of their real counterpart. An object-oriented approach generally includes four concepts: identity, classification, polymorphism, and inheritance (Rumbaugh et al., 1991).

Identity refers to the quantification of data as discrete objects. Objects can represent both concrete entities such as a reservoir, or concepts such as a reservoir operating policy (Rumbaugh et al., 1991).

Classification refers to the grouping of objects with the same properties and methods into a class. The class defines the properties and methods for the objects, with each object representing an instance of the class. In a water resources application, an example of a class might be a reservoir, while the Ashokan Reservoir and Center Hill Reservoir would be examples of reservoir objects. Although each reservoir object contains the same properties, such as the

name of its managing agency, the values of the properties may differ (Rumbaugh et al., 1991).

Polymorphism means that different classes may implement the same behavior in different ways. For instance, a reservoir object might perform a flood routing operation differently than a river object. Polymorphism allows new classes to utilize existing operations without the need for rewriting code, as long as each new class contains the code it needs to handle the operation (Rumbaugh et al., 1991).

Inheritance refers to the hierarchical sharing of properties and methods among related classes. Properties and methods common to several types of objects can be grouped into a superclass, also known as a parent class. Subclasses, or child classes, can then inherit those properties and methods in addition to defining their own. For example, a waterbody could be modeled as a superclass, with subclasses of river, lake, and fishpond. Each subclass may have a fish count property, while the lake and fishpond classes may also define a surface area property. Some superclasses are useful for grouping properties and methods, but are never used to instantiate objects of their own. These classes are called abstract classes. By grouping common properties and methods into superclasses and then utilizing inheritance, repetition in a program is greatly reduced (Rumbaugh et al., 1991).

Encapsulation is another key concept crucial to a sound object-oriented design. Encapsulation means that the external properties and methods of an object (those visible to other objects) are separated from the implementation

details of the object, which are hidden from other objects. By internalizing the implementation details, a system becomes much easier to maintain. The designer can change the implementation (for instance to fix a bug or improve efficiency) of a particular object's methods without having to change the way those methods are called by other objects (Rumbaugh et al., 1991).

2.2.2 COM

COM, which stands for Component Object Model, is a binary specification standard devised by Microsoft that allows compliant software to utilize the object libraries of other COM-compliant software. COM itself is not a programming language, although languages such as C++ and Visual Basic lend themselves towards COM-compliant software design. Rather, COM provides a standard set of rules for developing software such that components from a program with a COM-compliant design can access components from other COM-compliant programs, regardless of the language that each program was developed in.

COM may be most evident within the Microsoft Office applications of Excel and Word. Because each software can utilize the object libraries of the other software, each can incorporate useful components from the other software into its own documents. For instance, copying and pasting a range of Excel cells from a spreadsheet or a chart into a Word document is an easy operation and produces no error. In fact, Excel's charting capabilities are directly linked to Microsoft Graph, another COM-compliant object library distributed with Microsoft Office.

By incorporating COM-compliance into a software system's design, that software possesses the potential to utilize components from any other COM-compliant software. This means that a purely computational model could be extended to produce graphs, prepare reports, carry out spreadsheet operations, update databases, or even upload results to a web site, while keeping the core functionality of the model relatively simple.

2.2.3 Object-Oriented Programming and Software Design Principles

When combined with a COM-compliant design, object-oriented programming addresses many of the key software design principles stated above. Because object-oriented design focuses on the nature of the required objects and their relationship to their real counterparts, rather than just the functionality required by the current problem statement, object-oriented software possesses a greater potential for robustness that function-driven software. The modular nature of object-oriented programming lends itself to an extensible design, as changes in design requirements can be implemented by changing only the necessary objects.

Encapsulation further improves extensibility, as a change in an object's internal procedure is hidden from other components of the system, and thus the rest of the system needs no modification to support the object's internal change. A modular design and polymorphism promote reusability, by allowing reuse of both modules (objects) and operations when necessary.

Finally, a COM-compliant design and encapsulation allow components of an object-oriented system to be compatible with other programs, regardless of the programming language or implementation details of those components. One way

in which this is accomplished is through the incorporation of DLLs into a software system. A DLL, or *dynamic linked library*, is a set of objects, functions, or routines that operate in the same process space as the calling application. By including a DLL from another COM-compliant application in a particular application's software design, that application can use components from the other application that are included in the DLL.

2.3 INTERFACE DESIGN

A well-written computational model remains ineffective without a clear link between the program and the user. Graphical User Interface (GUI) design is one of the key issues in software construction today. Hartley (1998) provides seven useful design principles: User in Control, Directness, Consistency, Forgiveness, Feedback, Aesthetics, and Simplicity (Hartley 5).

2.3.1 User in Control

Users should feel that they are in control of the software. Navigation between different elements in the GUI should be relatively easy and straightforward. Selection processes should not make incorrect assumptions about which objects the user intended to select. Difficulty in routine navigation and selection tasks can quickly lead to frustration. The ability to customize the software should also be taken into account, as more experienced users often seek ways to maximize efficiency and improve functionality for specific applications. Keyboard shortcuts and Visual Basic for Applications macros can provide such customization potential (Hartley 1998).

2.3.2 Directness

The GUI should be intuitive to the user. If the user already possesses a sense of familiarity with the GUI, then the mental workload required by the user to learn the GUI will be reduced. Directness is evident in many programs that mimic the standard interface of Microsoft Windows applications. This interface includes File, Edit, View, and Help menus, and a toolbar with icons for opening, saving, and printing files (Hartley 1998).

2.3.3 Consistency

Consistency between different components of an application reduces the mental workload of the user by allowing the user to transfer experience between components. Elements of consistency include placement of controls, labeling of controls (font and name), and operational behavior (Hartley 1998).

2.3.4 Forgiveness

The application should be designed to handle errors resulting from either physical mistakes (an inadvertent key press) or mental mistakes (calling the wrong task.) The application should gracefully recover from such errors, and provide feedback about the nature of the error to assist the user in recovering from the error and preventing future mistakes (Hartley 1998).

2.3.5 Feedback

An interface should provide feedback for each action that a user takes. When a change is made to an object in the application, the interface should indicate that the request for change was accepted and that the operation was

successful. Excessive delays cause stress for the user, even if the delay is a matter of seconds. For long processing tasks, the application should indicate the system's status and progress. This is commonly accomplished with a progress bar (Hartley 1998).

2.3.6 Aesthetics

The layout of elements of a user interface should not detract from their function. In fact, proper grouping and aligning of elements will help to make the purpose of the interface more obvious. The elements should be arranged in a logical order (if applicable), with broader information at the top of the interface and more specific information proceeding down the interface (Hartley 1998).

2.3.7 Simplicity

A form should not prompt or show the user for an excessive amount of information. Rather, the information should be communicated in incremental forms. The elements of the interface should be arranged so that the interface is easy to use. Descriptive labels help clarify the purpose of a particular element. Grouping and utilizing different colors or font styles helps to distinguish different logical groupings of elements. For instance, a set of buttons that are related to a common task could be organized into a toolbar or menu (Hartley 1998).

2.4 EXISTING MODELS

The principles behind intelligent software construction, object-oriented programming and GUI design provide useful guidelines in developing a computational model on top of the ArcGIS Hydro data model. Examining how other existing hydrologic and hydraulic models incorporate those principles into

their design is an insightful and worthwhile task. Several models have been investigated for that purpose, with the nature of the models ranging from an extension of a GIS to fully independent programs.

2.4.1 HEC-HMS

The Hydrologic Modeling System (HMS), created by the US Army Corps of Engineers Hydraulic Engineering Center (HEC), is a hydrologic model that utilizes seven objects to route water through the landscape: Subbasin, Reach, Reservoir, Junction, Diversion, Source, and Sink. Each object is defined by the manner in which it conducts water through the landscape. All objects possess an identification number. The connectivity between objects is known, and is established for each object by storing the ID number of the next downstream object. From this, a schematic can be drawn of connected components. In HMS, water can only be passed in the downstream direction (no backwater effects). Thus, the objects in HMS act essentially as time series processors that receive an inflow time series from upstream sources, process the inflow to produce an outflow time series, and pass the outflow to the next downstream object (HEC, 2001).

A simulation run is governed by three major components: the Basin Model, the Meteorological Model, and the Control Specifications. The Basin Model contains information about the properties and connectivity of the seven objects described above. The Meteorological Model contains rainfall and evaporation information in the form of time series associated with rainfall gages.

The Control Specifications component defines simulation properties such as time step and duration (HEC, 2001).

The HEC has recently produced a library of routines used for hydrologic calculations in the form of a DLL called libHydro. These are many of the same routines used in the HEC-1 software (which is the predecessor to HMS), and thus have been proven through years of use. These routines can be accessed through calling statements in programming languages such as Visual Basic or C++ (HEC, 1995). By releasing HEC-1 routines in a reusable format, the HEC has made a vast library of useful hydrologic functions available.

2.4.2 Kortflom

Alfredsen and Saether (2000) created a program called Kortflom for performing flood analysis in river systems at the Norwegian University of Science and Technology. The model routes water through a river system that may include lakes and reservoirs. The object-oriented framework behind the program supports a set of hydrological components, topological relationships between those components, and specialized system behavior during simulation. The model is written in C++ and is designed to allow users to easily create new components, either through specification or generalization of existing components. To promote a robust design, the model allows for the use of different methods in calculating flows on reaches, depending on data availability and reach characteristics. The model is designed so that the basic structural components (e.g. rivers, lakes and catchments) of a real world river system are modeled along with their topological associations, but without a link to a specific computational

component. This makes it easier to choose which computational method to use for a specific component (Alfredsen and Saether, 2000).

The model is divided into four major components: structural components and system topology, computational methods, data containers, and simulation control. A special control object ensures that only appropriate components (such as a river branch) can be inserted at a given location. The model incorporates two time series classes, RegularTimeSeries and IrregularTimeSeries, with both classes inheriting from an abstract base time series class. RegularTimeSeries keeps data for time series with regular time intervals, while IrregularTimeSeries keeps data for time series with irregular time intervals. In addition to the classes designed to store time series data, several classes have been developed to handle time series transformations. These classes support transformation, sorting, and statistical analysis of time series data. During simulation, timing in the model is handled by a global time control class. Before a simulation can be run, three steps must be performed: 1) create the components to model the river system, 2) collect data to describe the components, and 3) connect the components in a topological network. In the future, the system may be stored in an object-oriented database (Alfredsen and Saether, 2000).

2.4.3 SMILE

Spanou and Chen (2000) developed an object-oriented tool called SMILE for modeling point-source pollution in river systems. The software allows the user to represent river basins, while supporting the computation of river flows and water quality.

To represent river basins, the software uses a Watershed object that is a composition of a RiverSystem object, and optionally of several other objects such as Community and WastewaterTreatmentPlant. The Watershed object is designed so that it can be easily extended to allow the inclusion of other objects (such as agricultural areas) if needed (Spanou and Chen, 2000).

The network in the model is made up of RiverNode and RiverReach objects. The RiverNode class has subclasses that represent flow change points (WasteDischargePoints) and monitoring points (SamplingStations). The RiverNode class also has a subclass called MixingPoints where computations involving flow continuity and mass balance are performed (Spanou and Chen, 2000).

2.4.4 Noah 1D

Murray and Kutija (2000) developed an object-oriented model called Noah 1D that uses a variety of numerical schemes to solve one-dimensional free surface and pressurized flow problems. The model uses a central ModelControl object to create and perform maintenance on each component of the model. A separate SolutionControl object actually performs the simulation process. Data input objects in the model consist of ConnectionEdges, ConnectionVertices, PhysicalEdges, CrossSections, and BoundaryConditions. ConnectionEdges and ConnectionVertices define how the components of the model are connected. PhysicalEdges contain data that describe the physical properties of an edge, or river segment. CrossSections define the geometry along an edge, while BoundaryConditions define the boundary conditions for the model. By utilizing

both ConnectionEdges and PhysicalEdges, the connectivity between components in the flow system is separated from the physical description of the components. Input data and solutions are also separated into different objects. While input data is maintained by the data input objects described above, solutions from simulations are stored in SolutionPoint objects (Murray and Kutija, 2000).

2.4.5 Map-Based Surface and Subsurface Flow Simulation Models

Ye (1996) developed a surface and subsurface flow simulation model that integrates hydrologic process calculations, a map-based representation of the hydrologic system, and the underlying database of information that describes the system. The application runs from within an ArcView 3.2 project file, with additional functionality implemented through code. Ye used an object-oriented approach in developing river basin and river section classes, with each class containing properties to describe an object's state, and simple behaviors to handle such tasks as drawing and returning spatial location. Behaviors associated with hydrologic and hydraulic processes were left to be described by mathematical models within the program (Ye, 1996).

To store time series data, Ye investigated creating a time series table for each parameter at each location. Thus three monitoring points measuring stage and flow velocity would be associated with six time series tables. However, Ye chose not to implement this strategy due to the large number of tables that would be created with even a small number of spatial features. Instead, Ye's application creates a time series table for each time series attribute. The columns in the table represent the spatial locations associated with the attribute, while the rows

represent the timestamps. Thus a cell in the table would represent the value of a time-varying attribute at a particular location at a particular time (Ye, 1996).

2.4.6 ArcFM

ArcFM (Arc Facilities Manager) is a standalone software application designed to facilitate the planning and management of water, electric, and gas utilities. ArcFM is more of a planning and analysis model, rather than a computational model, with an emphasis on the layout, status, and relationships between utilities features. The application consists of a GIS-based user-interface through which users can map and analyze utilities. The GIS is directly linked to a geospatial relational database, such as Oracle. ArcFM is built to work on the Windows NT platform, and was written with Visual Basic 5.0 (ESRI, 1998).

ArcFM is built on the ArcGIS software system. ArcFM provides the user with several features designed to streamline utilities management. These include a graphical user-interface with a custom ArcFM Viewer, a rich toolset geared towards facilities-based tasks (such as traces, mapping, and construction), and a RuleBase Engine (RBE) that supports automatic validation checks, connectivity assurance, and other business rules defined by the user (ESRI, 1998).

ArcFM's user-interface was designed with Visual Basic to resemble typical Windows-based applications. The software contains standard menus such as a "File" menu, and utilizes buttons and tools like MS Word or Excel (ESRI, 1998). Using familiar components in a graphical user-interface shortens the learning curve for a given application. This is an example of Directness as described by Hartley (1998).

ArcFM comes with its own viewer software. The viewer provides utilities for reporting, charting, mapping, and outputting data. With the viewer, the user can select from a list of feature types to create in a toolbox-style window. The viewer also contains custom tools such as an attribute inspection tool and a tool which assigns images to particular features (ESRI, 1999).

ArcFM contains a RuleBase Engine (RBE) to help insure model integrity and ease maintenance efforts by encoding business rules specified by the user. The RBE controls display of features, defines the structure of the user interface, asserts connectivity rules between features, and performs attribute validation for features. The RBE can simplify editing tasks by allowing only the appropriate pipes to be connected to certain valves, for example. When making a connection, the RBE automatically snaps features together, enforcing topological connectivity. By separating the rulebase from the actual data, ArcFM preserves the integrity of the data while enhancing the maintenance and creation of ArcFM features (ESRI, 1999). In addition to the basic rules provided by ArcFM, the user can add his own business rules to suit the needs of his specific application (ESRI, 1998).

An example of inheritance can be found in the ArcFM object model. To build the analysis model for ArcFM, individual classes were conceptualized. Properties of each class were then defined, in the process revealing common properties between classes. Higher-order classes were then created to define the common properties, with the appropriate classes inheriting from these superclasses. These superclasses were often abstract (ESRI, 2000).

2.5 CONCLUSIONS

Alfredsen and Saether, Murray and Kutija, and Ye each incorporated a differentiation between data and analysis operations by separating the structure and topology of the river system from computational components. Murray and Kutija further specialized river system components by storing connectivity (topology) between components and the physical description of the components in separate objects, which is similar to the use of HydroEdge (for connectivity) and HydroFeature (for properties) objects in the ArcGIS Hydro Data Model. This differentiation helps to secure the integrity of the data while promoting a modular structure in the software's design. This scheme also encourages use of preexisting computational procedures, such as those available in HEC's libHydro. Using proven routines not only shortens development time for a computational model, but also validation time as well.

The three steps that must be followed before a simulation is run in Kortflom are equivalent to the data preparation that routinely occurs in a GIS. The network model in SMILE bears a resemblance to the current network model in ArcGIS. In particular, the RiverNode subclasses of WasteDischargePoint and SamplingStation demonstrate a direct correlation to the WaterDischarge and MonitoringPoint classes in the ArcGIS Hydro data model. From these examples, it is clear that GIS and the ArcGIS Hydro data model may provide a sound data structure and pre-processing environment for the data components of a hydrologic simulation model.

Both Kortflom and Noah 1D use a simulation control object to control the simulation run, as well as objects to manage time series data. To keep track of the movement of water through the landscape, both spatially and through different processes, a hydrologic simulation model might benefit from a similar control object, like a conductor to the orchestra of hydrologic processes. While the ArcGIS Hydro data model does provide a TimeSeries class, a more elaborate scheme may be required to store time series information from a simulation model.

Each of the independent programs described above possesses the freedom of a structural and user-interface design best suited for its model needs. Each of those models can also run in the absence of a GIS system. However, ArcFM and Ye's simulation models, which are integrated into the GIS environment, have the benefit of using a GUI already familiar to GIS users. By creating a model that runs inside of a GIS, much of the work in developing a GUI can be avoided since a basic GUI already exists. ArcGIS's COM-compliant design and extensive object library also encourage development from within a GIS. Ultimately, the nature of the model and the resources available will probably determine the platform on which the model is built.

Chapter 3: Methodology

The new capabilities made possible by the latest innovations in GIS technology were one of the primary factors responsible for the development of the object-oriented ArcGIS Hydro data model (Davis, 1999). This chapter begins by discussing how hydrologic modeling lends itself to object-oriented programming, followed by a brief description of innovations behind ArcGIS's software design. It also describes the ArcGIS Hydro data model, including how it benefits from the GIS innovations and how it extends the capabilities of GIS software in a water resources engineering context. With a basic understanding of Arc Hydro and the software established, this chapter discusses approaches for developing a computational model on top of Arc Hydro. Finally, this chapter outlines some design alternatives for creating a user interface for the computational model.

3.1 Object-Oriented Programming and Hydrologic Modeling

The hydrologic cycle involves several processes, such as rainfall/runoff and channel routing, which must be integrated to produce a complete solution through modeling. Thus hydrologic and hydraulic modeling lends itself to a modular style of programming, in which each of the processes is controlled by a sub-model that is interconnected to other components in the whole system (Alfredsen and Saether 2000). Due to the modular nature of object-oriented programming, and the fact that object-oriented programming has now been incorporated into GIS technology through ArcGIS, it seems logical that a hydrologic simulation model would benefit from an object-oriented design.

Several modeling programs already exist that use this approach, such as HEC-HMS, Noah1D, and the Map-Based Surface and Subsurface Flow Simulation Models developed by Ye.

In terms of GIS Features, the basic identity and attributes of objects (the Features) are already defined. What remains is to associate methods or behaviors with those objects. The HEC has contributed a great resource to this effort by creating libHydro, a DLL containing the hydrologic functions and subroutines from the HMS software in a form that may be called by Visual Basic, C++, or other programming languages (HEC, libHydro). By calling these routines from an object's methods, an object can possess a variety of hydrologic behaviors such as runoff determination or channel routing. On the developer's side, development time is greatly shortened since the routines are already written. Validation time is also greatly reduced since the routines have been tested through years of use in the HEC software.

Because an object-oriented design lends itself to hydrologic modeling, and because object-oriented programming has been integrated into ArcGIS, this paper discusses approaches of computational model development from within the context of object-oriented programming.

3.2 SOFTWARE OVERVIEW

ArcInfo 8 and ArcView 8 are the latest releases of ESRI's GIS software. Collectively known as ArcGIS, the software exhibits many new features and innovations that mark it as the next step in the evolution of GIS. ArcInfo 8 is

ESRI's enterprise GIS software and is composed of three major components: ArcCatalog, ArcMap, and ArcToolbox.

ArcCatalog resembles Windows Explorer in look and feel, its purpose being the creation and maintenance of GIS files. ArcCatalog provides three types of views for browsing data: Contents, Preview, and Metadata. The Contents view shows tables and geodatabase files in the same way that folders and files are shown in the windows explorer (the geodatabase model is described in section 3.2.2). ArcCatalog can also display coverages and shapefiles, which are the file formats used by earlier versions of ESRI's ArcInfo and ArcView GIS software, respectively. The Preview view displays a preview of the highlighted GIS file, either through a spatial display or by showing the attributes of the data in a table. The Metadata view reveals metadata about the GIS file, such as the creator of the data, the date the data were created, etc.

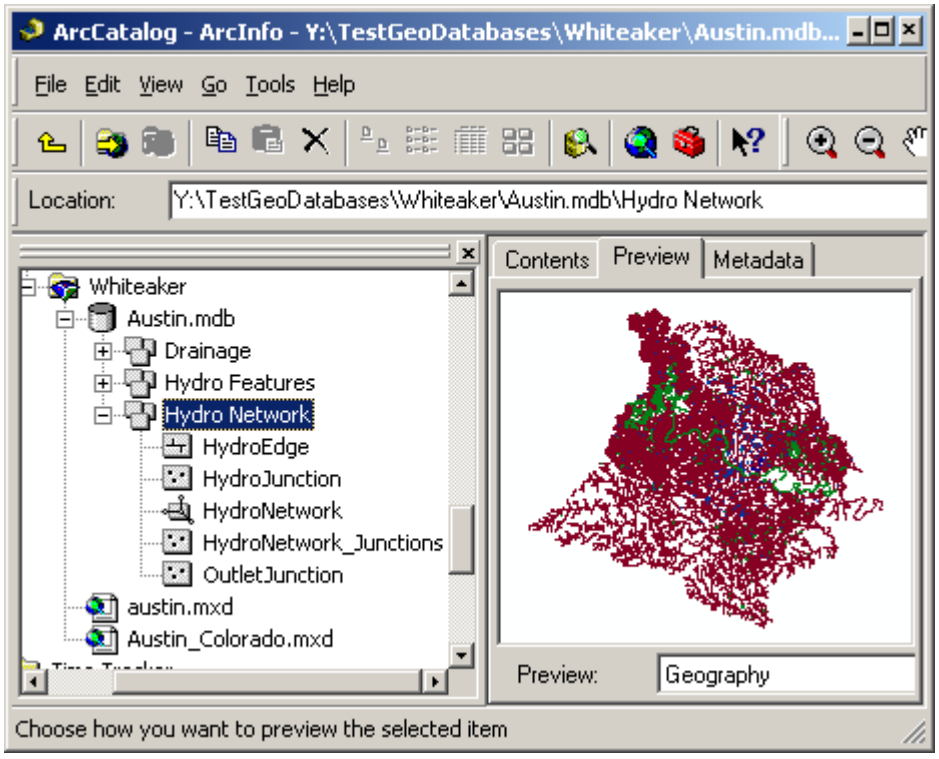

Figure 3.1 ArcCatalog Graphical User Interface

ArcCatalog is the utility used to create relationships between GIS features and to build geometric networks. Relationships define associations between features through attribute keys. Geometric networks define connectivity between line and point features. ArcCatalog can also be used to import and export GIS data from one file format to another.

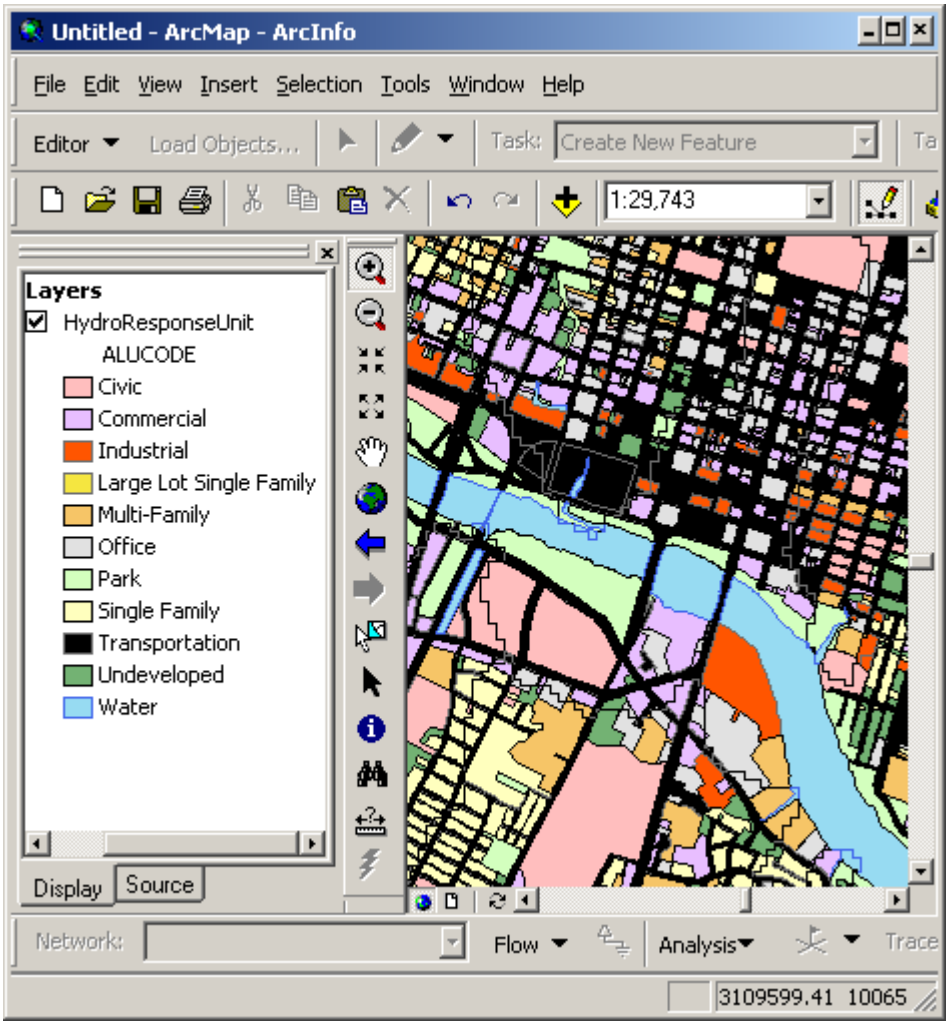

Figure 3.2 ArcMap Graphical User Interface

ArcMap is used for viewing and analyzing GIS data. It contains tools for zooming, panning, and making printable maps, as well as special editing and network tracing tools. Tools performing similar functions are organized into toolbars, which can be moved and docked in different locations to improve the ergonomics of the interface. The ArcMap user interface may be customized by

creating custom tools and toolbars, and then associating those tools with code that performs a particular task.

ArcToolbox contains many utilities that operate on GIS data, including projection utilities and data conversion tools. The data conversion tools use the same routines that ArcCatalog uses for importing and exporting data between different GIS file formats (e.g. coverages, shapefiles, etc.) GIS data can also be processed to create new data with ArcToolbox. For instance, the Create Centerlines tool creates a centerline between dual-line features.

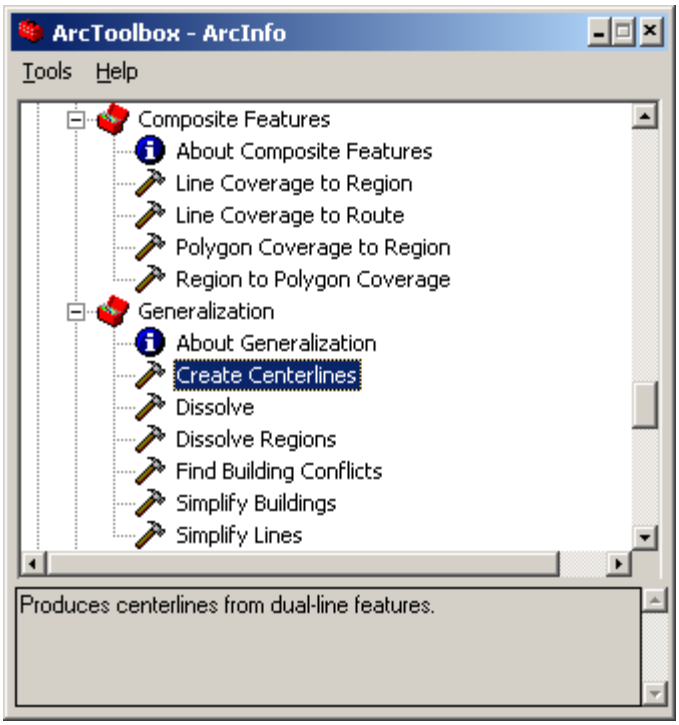

Figure 3.3 ArcToolbox Graphical User Interface

ArcView 8.1 is sold as a separate package from ArcInfo 8 and is used for viewing GIS data. ArcView 8.1 can edit shapefiles and simple feature classes, but it cannot create geometric networks or edit network feature classes.

3.2.1 RDBMS Technology

In previous versions of ESRI's GIS software, spatial data were stored in a proprietary format. ESRI attempted to use a more database-centric structure upon the release of their ArcView GIS software. Yet while ArcView versions 1-3 could store attribute information in dBase format, the geographic representation of data was still linked to a separate file.

This changed when ESRI incorporated relational database management systems (RDBMS) structure into ArcGIS. Previous database technology required fields in a table (such as a dBase table) to be of a standard data type, such as integer or string. For this reason, old versions of ArcView could store the attributes of a Feature in a dBase file, but the information necessary to define that Feature's spatial representation had to be stored in a separate file. However, today's RDBMS software allows a table to possess one or more blob fields. A blob field can store data of any type, including standard data types such as integers and strings, as well as images and files. ArcGIS utilizes a blob field in the design of Feature Classes. A Feature Class is a table made up of Features of the same type, such as rivers. Rows in the table represent an individual Feature, such as the Tennessee River. By including a blob field in the Feature Class table, the attribute information of the Tennessee River is stored alongside its spatial representation, providing a more intimate link between the two.

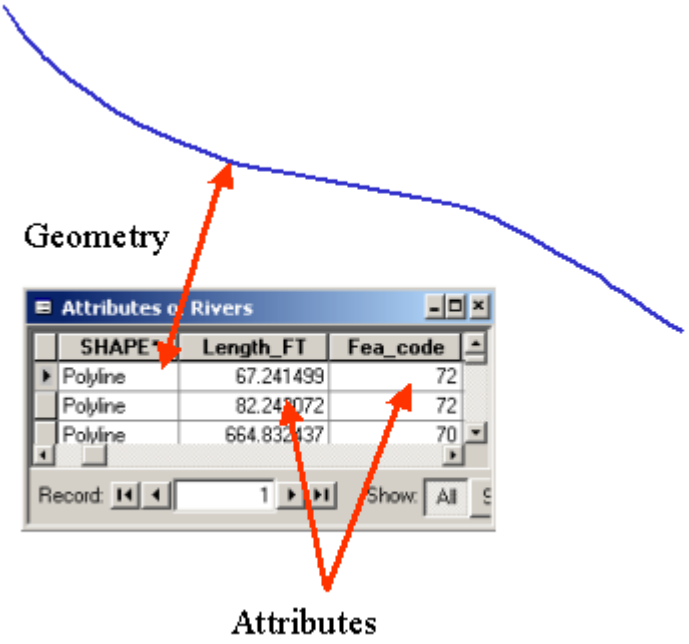

Figure 3.4 Feature Class Table Structure

3.2.2 Geodatabase Structure

ArcGIS supports two main categories of Features: Simple Features and Network Features. Simple Features include points, lines, and polygons. Network Features include Simple Edges, Complex Edges, Simple Junctions, and Complex Junctions. All Edges are connected by Junctions. A Simple Edge is a linear Network Feature with no internal junctions. A Complex Edge is a linear Network Feature that may contain one or more internal junctions, which are junctions that lie on the edge but do not split the edge. Thus a Complex Edge may join another Complex Edge anywhere along its length, while Simple Edges can only join other Simple Edges at their endpoints. Simple Junctions can be thought of as the nodes

that connect Edge Features, although Junctions do not have to be attached to any Edges. Complex Junctions are Junctions with special internal connectivity, analogous to a switchboard.

A collection of Features of the same type is stored as a Feature Class. Each row in the Feature Class table represents an individual Feature. Feature Classes that share a common use can be grouped into Feature Datasets. A Feature Dataset is a container that defines a reference frame for the Feature Classes that it contains. The reference frame includes information about the spatial projection, coordinate range and coordinate precision for the data. Feature Datasets can also store relationships between Feature Classes, as well as geometric networks. Relationships in ArcGIS are comparable to relationships in any RDBMS, with related rows in different tables being linked by a common identifier in key fields in each table. Geometric networks are used for defining network topology between Features. Geometric networks support tracing and connectivity tasks. Only Network Features may participate in a geometric network.

Feature Datasets, Feature Classes, relationships, geometric networks, and non-spatial tables are all stored in a Geodatabase. A Geodatabase is a relational database that serves as a container for spatial data in ArcGIS. Other RDBMS software, such as Oracle or Access, can open a Geodatabase. Using such software to view a Geodatabase reveals Feature Class tables, as well as other tables used to maintain the Geodatabase.

3.2.3 Custom Features

ArcGIS has extended the power and functionality of a Feature by incorporating object-oriented technology into its software design. In addition to their spatial and attribute information, Features can also possess special behaviors through the use of interfaces. For example, in addition to being a simple blue line on a map, the GIS representation of a river may also know how to route a flood wave from its upstream to its downstream end, how to draw itself at different scales, and which Features to notify if its spatial or attribute information changes. By adding custom behavior to Features, the GIS representation of real-world objects becomes a more accurate depiction of the reality that GIS is trying to model.

Custom Features can also take advantage of ArcGIS's COM-compliant design. Because ArcGIS is COM-compliant, Features can access the capabilities of software such as Microsoft Excel to plot graphs, or Word to prepare reports. The code behind a Feature's behavior can be written with a COM-compliant programming language, such as C++, meaning that users no longer must learn a proprietary programming language to customize the software.

To create custom Features, the name, inheritance, attributes, and interfaces are created in the Unified Modeling Language, or UML. UML is a standard language for writing software blueprints using object-oriented techniques (Booch, Grady, James Rumbaugh, and Ivar Jacobson, 1999). Custom ArcGIS Features are created in a UML static structure diagram, which is analogous to an object model diagram. As defined by Rumbaugh et al., an object model "describes the

static structure of the objects in a system and their relationships" (6). Each diagram in the model contains nodes, which represent custom Feature or Object classes, and arcs, which represent relationships (inheritance, associations, etc.) between classes. Packages organize UML items to display a common role of a set of classes, or to show relationships between classes.

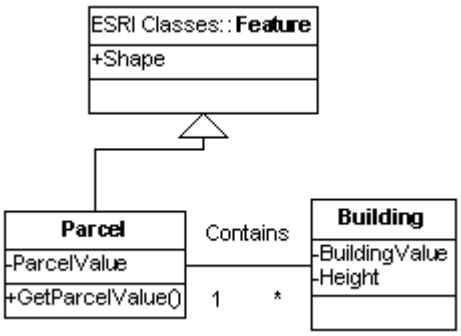

Figure 3.5 Sample UML Diagram

In the diagram above, the custom Parcel class inherits from the ESRI class Feature, as designated by the line with the arrow pointing from Parcel to Feature. In addition to Parcel inheriting the Shape attribute of Feature, Parcel also defines a ParcelValue attribute and a GetParcelValue method (behavior). Although a method has been defined in the static structure diagram, the implementation of the method actually occurs in the Visual C++ or other programming environment. Parcel is in a one-to-many relationship with the custom Building class. The relationship is read as Parcel Contains Building. Each object instantiated from the Parcel class may contain one or more Building objects.

UML diagrams are created with a CASE (Computer Aided Software Engineering) tool, which is a graphic software system. Once the object model has been created with a CASE tool, the UML is exported to the Microsoft Repository. Using a code generation wizard that ships with the ArcGIS software, stub code can be generated in Visual C++ from the repository, and the code required to implement each of the behaviors of the geoobjects can be created. After code generation is complete, a DLL is created and linked to all instances of the custom Features in ArcGIS. By accessing the Microsoft Repository in ArcCatalog using the Schema Creation Wizard, Feature Classes that will store custom features can be created, or the schema can be applied to existing data in a geodatabase.

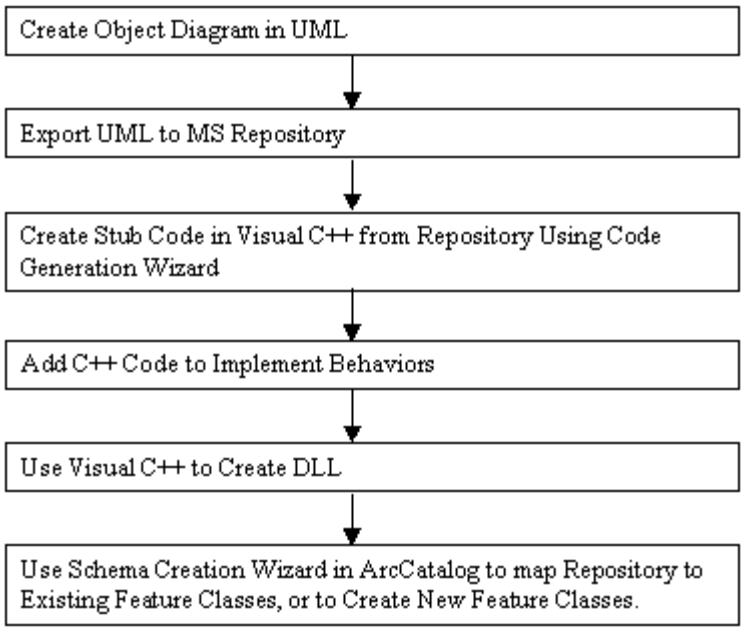

Figure 3.6 Procedure for Creating Custom Features in ArcGIS

3.2.4 COM-Compliance

ArcGIS is the first GIS software released by ESRI with a COM-compliant design. Through COM, ArcGIS can now communicate with other COM-compliant software, such as Word, Excel, and Internet Explorer, by utilizing public components (those that can be accessed by other applications) from each software's object library. ArcGIS also uses Visual Basic for Applications (VBA) as its customization language, no longer requiring users to learn a proprietary language (such as Avenue or AML) for customization purposes. Note that Visual Basic (VB) is different from VBA, in that Visual Basic is used to create standalone applications or DLLs, while VBA is used from within a software application to customize that application. Through VBA, the graphical user interface of ArcMap and ArcCatalog can be tailored to the fit the needs of the user. Custom buttons and toolbars can be created, and macros can be written to automate complex tasks. VBA is also the means by which the object libraries of other COM-compliant software are accessed. This means that ArcGIS can now link spatial data to spreadsheet applications, reports, and even web utilities. The customization potential provided by COM-compliance and VBA has extended the functionality of ArcGIS far beyond that of any GIS software in the past.

3.2.5 MapObjects

In addition to the customization capabilities within ArcGIS, ESRI has also provided a means to create separate programs that utilize one or more components from ArcGIS. These streamlined applications incorporate only the ArcGIS components they need, resulting in a much smaller and much faster software

product than the powerful ArcGIS software. This collection of reusable components is referred to as MapObjects.

MapObjects consists of a Map control and over thirty other objects that can be used in Visual Basic, Delphi, and other industry standard programming environments (ESRI 8). These controls possess much of the same functionality found in ArcGIS. Applications built with MapObjects can support the display of spatial information, with functionality such as panning and zooming. MapObjects also supports basic querying of features (either spatially or by attribute data), location of addresses, feature selection, and statistical calculations. While MapObjects is not intended to act as a substitute for complete ArcGIS functionality, it can add GIS capabilities to an application that would otherwise be lacking a mapping component.

3.3 ARCGIS HYDRO DATA MODEL

The ArcGIS Hydro data model attempts to take advantage of some of the new customization capabilities available in ArcGIS. The ArcGIS Hydro data model is a collection of custom feature and object classes organized to facilitate the preparation and maintenance of geospatial data for use in water resources applications. It is designed to work within the ArcGIS software system. Arc Hydro itself is not a simulation model. The primary goal of Arc Hydro is to provide a schema that, when applied to GIS data, produces data that can be easily used in hydrologic or hydraulic analyses. In essence, it is a data support system for hydrologic simulation. Yet in organizing Arc Hydro's classes, a structure for developing GIS data useful in a water resources engineering context was also

developed that is of value even if Arc Hydro itself is not applied. Arc Hydro serves as a conceptual framework that integrates the latest advances in GIS technology with the latest advances in the realm of hydrologic database development.

The ArcGIS Hydro data model was designed in UML using Visio 2000, a CASE tool. Arc Hydro is composed of five major packages, with four packages representing Feature Datasets that will be included in the geodatabase, and the fifth package representing time series data. The packages are Hydro Features, Hydro Network, Drainage Areas, Channel Features, and Time Series.

3.3.1 Hydro Features

The Hydro Features package contains twelve classes used to represent the cartographic features of the landscape. The HydroPoint, HydroLine, HydroArea, and Waterbody classes store typical hydrography data layers such as rivers, swamps, and lakes. In addition to map hydrography, there is a vast collection of water resources features stored in extensive databases, such as the National Inventory of Dams (U.S. Army Corps of Engineers par. 1). The Dams, Bridges, Structures, WaterWithdrawal, WaterDischarge, MonitoringPoint and UserPoint classes are designed to store such features. Each of these classes inherits attributes from an abstract HydroFeatures class. These attributes include HydroID, an identifier for features in Arc Hydro; and FeatureType, which is used to further distinguish features for cartographic or display purposes. The Hydro Features package also contains a HydroResponseUnit class, which is used to represent data pertaining to the calculation of the vertical water balance. This class contains

such data as the distribution of soil types, climate data, land use, and administrative boundaries.

Class (Type)	Inherits From	Attribute (Type)	Description
Dam (Point)	HydroPoint		A structure creating a pond or reservoir storing water
Bridge (Point)	HydroPoint		A structure carrying a road across a stream
Structure (Point)	HydroPoint		Any other kind of water resources structure
Monitoring Point (Point)	HydroPoint		A measurement station or sampling point
Water Withdrawal (Point)	HydroPoint		Point of withdrawal of water
Water Discharge (Point)	HydroPoint		Point of discharge of water
UserPoint (Point)	HydroPoint		Any other point of interest
Hydro Feature (Abstract)	Feature (ESRI)		Abstract class with common attributes and methods for Hydro Features
		HydroID (String)	Unique feature identifier in the geodatabase
		FeatureType (String)	Type of geographic feature
		Name (String)	Geographic name
HydroPoint (Point)	HydroFeature		Point features from map hydrography and inventory sources
		JunctionID (String)	Identifier for the corresponding junction on the network
HydroLine (Polyline)	HydroFeature		Line features from map hydrography
HydroArea (Polygon)	HydroFeature		Area features from map hydrography

Waterbody (Polygon)	HydroArea		An area of water
		AreaInSqKm (Double)	Area independent of map units
		JunctionID (String)	Identifier for the junction at the outlet of the area
Hydro ResponseUnit (Polygon)	Feature (ESRI)		Any subdivision of the landscape used for surface water balance accounting
		HydroID (String)	Unique feature identifier in the geodatabase
		AreaInSqKm (Double)	Area independent of map units

Table 3.1 Hydro Feature Classes

3.3.2 Hydro Network

The Hydro Network package contains a geometric network named HydroNetwork designed to transmit water through the drainage system of the landscape. The network includes a HydroJunction class, which is used to connect edges to edges or drainage areas to the river network, and a HydroEdge class, which transports the water along the network. Key attributes on HydroEdge include ReachID, LengthDownstream, HydroEdgeType, and FeatureType. ReachID is an identifier used for linear referencing events along a river such as a water quality segment. LengthDownstream stores the distance from the most downstream node of the edge to the sink (lake, ocean, sinkhole, etc.) to which the river flows. This attribute is useful in establishing a measure system for linear referencing, or for finding the distance between segments of a river. HydroEdgeType distinguishes the two major categories of HydroEdges: Flowlines and Shorelines. Shorelines represent edges along the boundaries of

areal water features, such as lakes or oceans. Flowlines represent all other edges including river centerlines and streamlines through a pipe system. FeatureType provides an additional level of classification for HydroEdges.

The HydroNetwork is associated with two classes designed to support linear referencing on the network, HydroPointEvent and HydroLineEvent. These classes store events at a single point along on edge, and across a segment of one or more edges, respectively. These classes inherit from an abstract HydroEvent class, which contains a ReachID attribute that maps to the ReachID attribute in HydroEdge.

Class (Type)	**Inherits From**	**Attribute (Type)**	**Description**
HydroEdge (Complex Edge)	Complex Edge (ESRI)		Linear segments in the HydroNetwork
		HydroID (String)	Unique feature identifier in the geodatabase
		ReachID (String)	Reach identifier (used for linear referencing)
		Name (String)	Geographic name
		LengthInKm (Double)	Length independent of map units
		Length Downstream (Double)	Length along shortest path to a downstream reference location
		Flow Direction (Integer)	Labels flow direction: Uninitialized = 0, WithDigitized = 1, AgainstDigitized = 2, Indeterminate = 3
		FeatureType (String)	Type of geographic feature
		HydroEdge Type	Type of HydroEdge: Flowline = 1

		(Integer)	Shoreline = 2
HydroJunction (Simple Junction)	Simple Junction (ESRI)		Junctions in the HydroNetwork, used for outlets, sinks, or other purposes
		HydroID (String)	Unique feature identifier in the geodatabase
		Next Downstream (String)	Identifier for next downstream feature in the HydroNetwork
		Length Downstream (Double)	Length along shortest path to a downstream reference location
		Drainage Area (Double)	Accumulation of all areas that drain to this junction, independent of map units
		FeatureType (String)	Type of geographic feature
HydroEvent (Abstract)	Object (ESRI)		Abstract class with common attributes and methods for events
		ReachID (String)	Identifier of linear referencing segment, maps to ReachID on HydroEdge
HydroPoint Event (Object)	HydroEvent		A point event
		Measure (Double)	Measure location of an event
HydroLine Event (Object)	HydroEvent		A line event
		FromMeasure (Double)	Measure location of the start of the line event
		ToMeasure (Double)	Measure location of the end of the line event
		Offset (Double)	Offset distance that the event is displayed from the HydroEdge

Table 3.2 Hydro Network Classes

3.3.3 Drainage Areas

A drainage area is an area of the landscape that drains to a point on a river network, to a river segment, or to a waterbody. The Drainage Areas package contains five classes that represent the elevation-based drainage pattern of the landscape: DrainagePoint, DrainageLine, Catchment, Watershed, and Basin. Catchments represent drainage areas defined by a consistent set of rules, such as a threshold drainage area method or the Pfaffstetter coding system. Watersheds represent any arbitrarily defined drainage area. Basins refer to a set of watersheds administratively derived to represent the principal drainage areas in a particular region, such as the 8-digit Hydrologic Cataloging Units established in the United States. Drainage lines represent the primary courses of water through the landscape as derived through digital elevation model (DEM) analysis. Due to the practical limits on the precision of elevation data, these lines often are not exactly spatially coincident with the actual river system. A drainage point lies at the center of a digital elevation model cell, which serves as the outlet cell of a DEM-derived drainage area. Drainage points are also known as seed points or pour points.

Class (Type)	Inherits From	Attribute (Type)	Description
Drainage Feature (Abstract)	Feature (ESRI)		Abstract class for drainage system features
		HydroID (String)	Unique feature identifier in the geodatabase
		DrainageID (Integer)	Link between point, line and area features of a drainage system, such as GridCode,

			Pfaffstetter number, or HUC number
Drainage Point (Point)	Drainage Feature		Point at the center of a DEM cell on a drainage path, usually seed point location for drainage area delineation
		JunctionID (String)	Identifier for the junction that corresponds to the drainage point
DrainageLine (Polyline)	Drainage Feature		Line through the centers of the DEM cells on a drainage path
DrainageArea (Abstract)	Drainage Feature		Abstract class for common drainage area attributes
		AreaInSqKm (Double)	Drainage area independent of map units
		Next Downstream (String)	Identifier of next downstream area in this drainage area Feature Class
		JunctionID (String)	Identifier for the junction at the outlet of the area
Catchment (Polygon)	DrainageArea		An elementary drainage area produced by a uniform process of landscape subdivision
Watershed (Polygon)	DrainageArea		Any subdivision of the landscape into drainage areas
Basin (Polygon)	DrainageArea		A set of standardized drainage areas for data archiving and delivery

Table 3.3 Drainage Area Classes

3.3.4 Channel Features

The Channel Features package contains three classes used to define the channel system of a river. The ProfileLine class defines longitudinal segments of the channel. This class contains three subtypes for further classifying the longitudinal components of a channel system: Thalweg, Bankline, and

Streamline. The thalweg represents the line formed by connecting the lowest points in each transverse section of the channel. The thalweg can also represent the channel centerline. Banklines represent the boundary of the channel at a given discharge. Streamlines represent any other flow lines in the longitudinal direction. The CrossSection class defines transverse sections of the channel. Both the CrossSection and ProfileLine classes contain an (x,y) spatial location, elevation, and measure values.

Sometimes historical cross-section data contain no (x,y) location information, but do contain river stationing location of the cross-section along the river. These data can be stored in the CrossSectionPoint class, which is a non-spatial (object) class containing cross-section events.

Class (Type)	**Inherits From**	**Attribute (Type)**	**Description**
Channel Feature (Abstract)	Feature (ESRI)		Abstract class for common channel attributes
		HydroID (String)	Unique feature identifier in the geodatabase
		ReachID (String)	Identifier of linear referencing segment, analogous to ReachID on HydroEdge
		RiverID (String)	Identifier of linear referencing segment, usually corresponds to named rivers
ProfileLine (3D Polyline)	Channel Feature		Longitudinal profile of the channel
		ProfileLine Type (Integer)	Labels profile lines: Thalweg = 1, Bankline = 2, Streamline = 3
CrossSection (3D Polyline)	Channel Feature		Transverse section of a channel
		Cross	CrossSection identifier

		SectionID (String)	
		CrossSection Origin (String)	Description of origin of cross section data
		ProfileM	Location of the CrossSection on ProfileLine's measure system
		JunctionID (String)	Identifier for the junction at the outlet of the area
CrossSection Point (Object)	Object (ESRI)		Non-spatial cross-section data
		Cross SectionID (String)	Identifier of the corresponding CrossSection feature
		Cross SectionM (Double)	CrossSection measure point location
		Elevation (Point)	Elevation of CrossSection point above mean sea level

Table 3.4 Channel Features Classes

3.3.5 Time Series

One of the most crucial components in hydrologic and hydraulic computations is time series data. The ArcGIS Hydro data model stores time series in a simple TimeSeries class with four attributes: FeatureID, TSType, TSDateTime, and TSValue. FeatureID is the HydroID of the feature related to a particular time series record. TSType, TSDateTime, and TSValue represent the type of time series data, the timestamp for a particular value, and the value, respectively.

3.3.6 Stage of Development

Currently, none of the Feature Classes in the ArcGIS Hydro data model possess behavior. Efforts to this point in Arc Hydro's development have been focused on establishing the Feature Classes necessary to represent the water resources domain in GIS, as well as organizing those Feature Classes into useful Feature Datasets. The next step in Arc Hydro's development is to determine what interfaces should be added to the feature classes to support the main goal of the model. Examining how Arc Hydro can be connected to a computational model may help determine some of those interfaces.

3.4 DEVELOPMENT APPROACHES

ArcHydro is a data model, or a model providing a structure for storing water resources data in a GIS. This is different from a water resources simulation model, which performs calculations on hydrologic or hydraulic data. There are two primary approaches for building a simulation model on top of the ArcGIS Hydro data model: Using Arc Hydro as a Pre-Processor, and Extending Arc Hydro. It should be noted that while each approach is different, elements of both could be merged to produce a blend of the two approaches.

3.4.1 Data Model Pre-Processing

The majority of water resources computational applications utilizing information prepared by a GIS involve pre-processing data in the GIS, and then exporting that data to a separate program where the bulk of the computations are carried out. For example, CRWR Pre-Pro is a set of GIS utilities developed at the Center for Research in Water Resources at University of Texas at Austin for

generating data, which can be imported into HEC-HMS for analysis (Olivera, 2001). The ArcGIS Hydro data model is intended to be used for this purpose, although on a much broader scale. Arc Hydro organizes features in a manner that lends itself to computational modeling. Arc Hydro also provides core attributes that often prove useful in hydrologic and hydraulic analyses, such as unique identifiers and measure values.

In some cases, the computational model is built within the GIS. This approach tends to avoid some of the errors or difficulties that may occur when attempting to communicate between two different software packages. In this scenario, the computational model could directly access features in the ArcGIS Hydro data model. Some basic concepts behind developing an internal or external computational model are discussed below.

3.4.1.1 External Model

Creating a water resources model that runs independently of ArcGIS provides the model developer with much freedom of design. In fact, this is the route that most hydrologic simulation models have taken. The disadvantage of this approach is that the developer may have to "recreate" some of the core functionality provided by ArcGIS, such as a network model and editing routines. While some of this functionality can be added using MapObjects components (viewing, querying, etc.), the more powerful GIS operations can take place only within an ArcGIS application. Of course, ideally this type of functionality would not be required, as the ArcGIS Hydro data model has created the bulk of the necessary spatially related information for the computational model.

Since the computational model is designed to run externally from the GIS, some routine would have to be developed to export data from the GIS to a format that the computational model can understand. Note that an export/import routine may not be required if the model utilizes the ESRI object library (which is possible given ArcGIS's COM-compliant nature); however, an external model should be expected to run in the absence of GIS data or even a GIS system, so the ESRI object library would most likely not be included in the model. This is the case with HEC-RAS, which allows the user to either create the components for the model simulation in the RAS user interface, or import the information required to create the components from exported GIS data (HEC, 1999). The simplest approach is to export to and import from a file format that is both compatible with and efficient to access from the GIS and the computational model. With ArcGIS, the two most obvious choices for file types are database files and text files. No matter which file format is used, the routines used to export the GIS data should be independent of Arc Hydro. Programming export routines that are compatible with every hydrologic or hydraulic model in existence into the basic structure of Arc Hydro would make Arc Hydro very cumbersome to use and maintain.

3.4.1.2 Internal Model

If a hydrologic model is built to operate within a GIS, then the problem of creating export routines is avoided since computational model components can communicate directly with Arc Hydro components. An internal model can also incorporate the functionality provided by the GIS. In this scenario, the model

would be created as an ActiveX DLL that utilizes the ESRI object library. The DLL can then be added to an ArcMap document as a custom tool. The disadvantage of this approach is that the computational model's operation must follow the rules of the ArcMap application. In other words, if the model tries to get too fancy, it may generate an error that causes ArcMap to crash.

3.4.2 Data Model Extension

Another approach to developing a computational model on top of the ArcGIS Hydro data model is simply to extend the core functionality of Arc Hydro. Algorithms already exist for using the structure of Arc Hydro to generate data that is useful in hydrologic or hydraulic analyses. In a relatively simple application, the classes in Arc Hydro could be extended one step further to include the capabilities of hydrologic computations. An extension of Arc Hydro has already been created to support Digital Flood Insurance Rate Maps (DFIRM) for the National Flood Insurance Program. This extension includes the creation of new classes that inherit from Data Model classes, as well as some simple methods involving parcel value computations (Donnelly, 2001). In a hydrologic extension to Arc Hydro, routing methods would be associated with linear components (HydroEdge, ProfileLine), while methods for calculating runoff would be associated with drainage area classes (Watershed, Catchment, Basin.) However, as the scope and computational needs of a hydrologic model grow more complex, so do the methods and components needed to perform the computations. A very complex hydrologic model would require so many modifications to the existing Data Model, that creating a computational model independent of Arc Hydro

would become a much cleaner approach. In other words, Arc Hydro should only be extended when the application is relatively simple and the structure of Arc Hydro already provides some core functionality required by the computational model.

3.5 INTERFACE DESIGN

There are three main alternatives for creating a user interface for a computational model in the context of ArcGIS. The first involves the creation of an external computational model. In this case, the developer has complete freedom as to the design of the interface. However, the developer should still follow guidelines of sound interface design, such as those outlined by Hartley (1998).

The second alternative is to create a custom graphical user interface within ArcGIS. This is the approach that ArcFM uses with its special ArcFM Viewer. The Viewer is built from the basic ArcGIS GUI components, with additional tools and buttons designed to work with the ArcFM software (ESRI, 1998). By developing the user interface within the GIS, many of the basic interface components (such as file menus, selection processes, etc.) from the GIS may be used, resulting in a shortened development time for a given application.

The third alternative is to operate the computational model from a custom toolbar or menu in the GIS. This approach works best when the model is relatively simple and does not require a robust set of tools and procedures to prepare a simulation run.

Chapter 4: Procedure of Analysis

Due to the great breadth involved in the procedure for creating software systems, that subject will not be discussed in this paper. There are several books, software packages, and courses available that are designed to assist the developer in the creation of new software, both within and independent of existing GIS platforms.

The process of extending the ArcGIS Hydro Data Model with new classes designed to perform hydrologic and hydraulic computations is also a subject that will not be discussed here. See Davis (1999) for a discussion of how to create custom features for use in ArcGIS.

For this research, eleven tools were created which operate on the Feature Classes in the ArcGIS Hydro Data Model to integrate the Feature Classes and provide support for more intensive hydrologic computations. These tools are implemented as custom add-ins in the ArcMap environment. The general procedure for creating these tools is described below, followed by a discussion of how the design principles of software construction and GUI design were applied in the development of the toolset.

4.1 Creating Tools for Use in ArcMap

By using the ESRI object library, a DLL can be created in Visual Basic that can be added to an ArcMap document as a custom tool. Custom tools extend the functionality of ArcMap to perform tasks specific to a user's needs. Eleven tools were created for this research. The general procedure used to create these

tools is outlined below. This discussion assumes a basic knowledge of the Visual Basic development environment and the ArcMap user interface. As an illustrative example, a sample tool is described that operates on the first layer in an ArcMap document. For each selected Feature from the layer, the tool displays a message box giving that Feature's ObjectID.

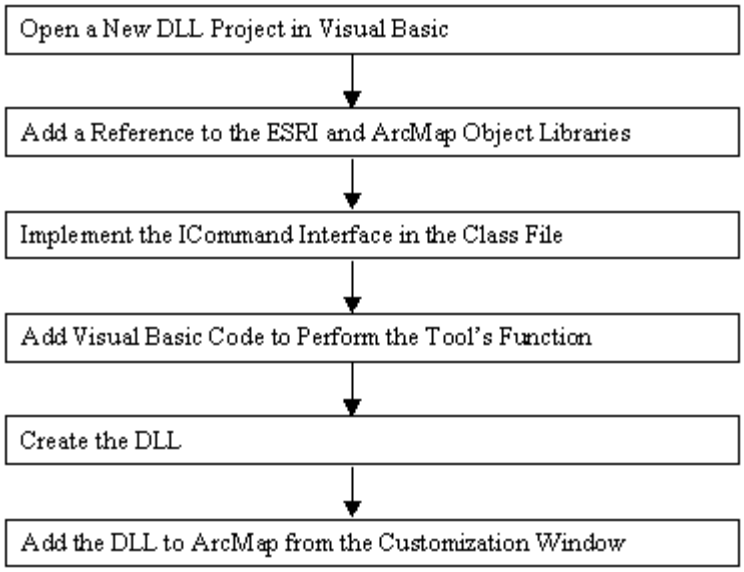

Figure 4.1 Procedure for Creating a Custom Tool in ArcGIS

4.1.1 Creating the DLL

Custom tools in ArcMap are implemented as DLLs that are created in Visual Basic. Note that Visual Basic is a separate software system from ArcGIS and must be loaded on the machine in addition to ArcGIS in order to create custom tools.

To create a new DLL, start Visual Basic by clicking Start>Programs>Microsoft Visual Studio>Microsoft Visual Basic. When prompted for the type of project to create, click ActiveX DLL and then click Open. Visual Basic prepares the current project for the creation of a DLL. In the Project Explorer window, note that a single class (Class1) has been created. Class1 is the default startup component for the DLL, which serves as the link between ArcMap and the functionality of the custom tool. In the Properties window, several properties for Class1 are listed. For the purposes of this discussion, Class1 is renamed to clsSampleTool. In the (Name) property of Class, type clsSampleTool and press Enter.

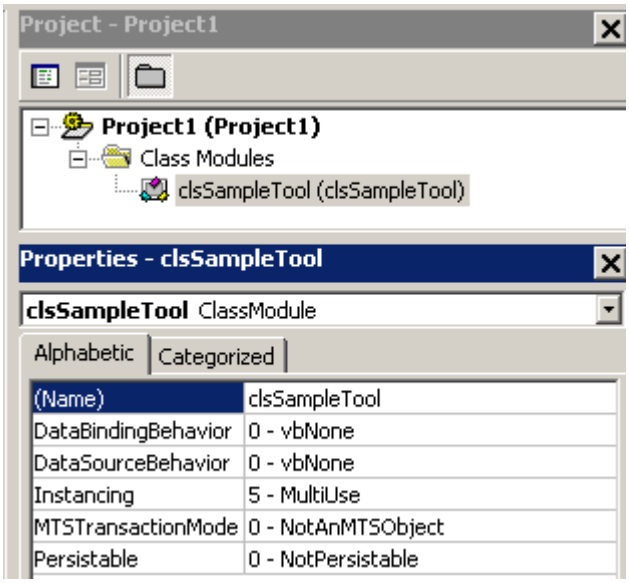

Figure 4.2 Properties of clsSampleTool

4.1.1.1 Implementing the esriCore.ICommand Interface

In order for ArcMap to recognize clsSampleTool as a custom tool, the clsSampleTool must implement the ICommand interface. ICommand is one of the COM interfaces available in the ESRI object library (see Fig. 4.3). The ESRI object library contains a standard set of object classes and interfaces used to perform various tasks in ArcGIS. A COM interface is simply a declaration of related properties and methods that may be used by a class. No implementation code exists in the interface. The implementation details are left up to the class that implements the interface. The properties and methods of ICommand are used to define a tool in ArcMap. Before the ICommand interface can be implemented, the project must obtain a reference to the ESRI object library. On the Project menu, click References to open the references window. Place a check by ESRI ArcMap Object Library and ESRI Object Library, and then click OK. The project can now access all public components of the ESRI Object Library.

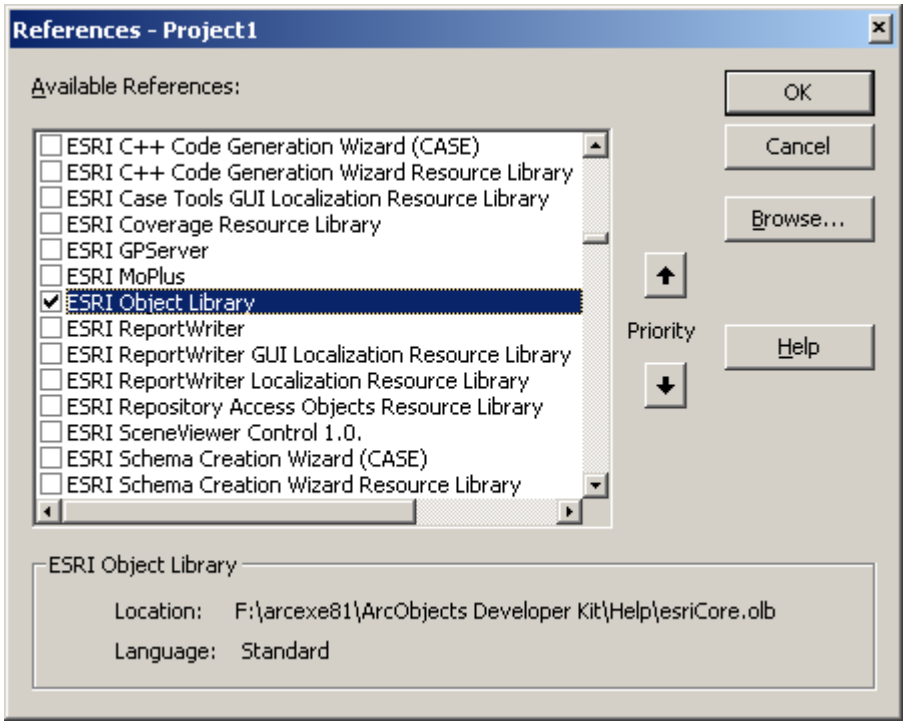

Figure 4.3 Adding a Reference to the ESRI Object Library

In the code window for clsSampleTool, add the following line of code.

```
Implements esriCore.ICommand
```

This line of code informs the project that clsSampleTool will implement, or supply the code for, the properties and methods of the ICommand interface in the ESRI object library. In the object drop down box in the code window, click ICommand.

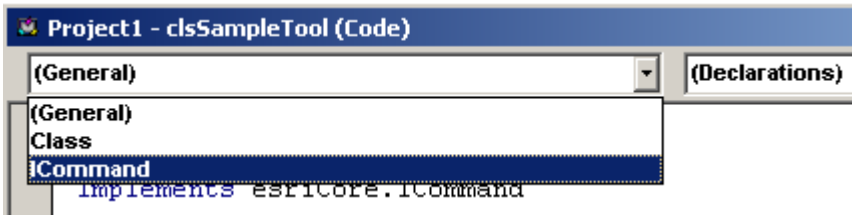

Figure 4.4 Implementing ICommand Interface

In the Declarations drop down box, notice that all of the properties and methods for ICommand are now listed. Click on each item in the drop down box to generate stub code for the properties and methods in the code window. Stub code provides the basic definition for a property or method, without defining how the property or method is accessed or implemented.

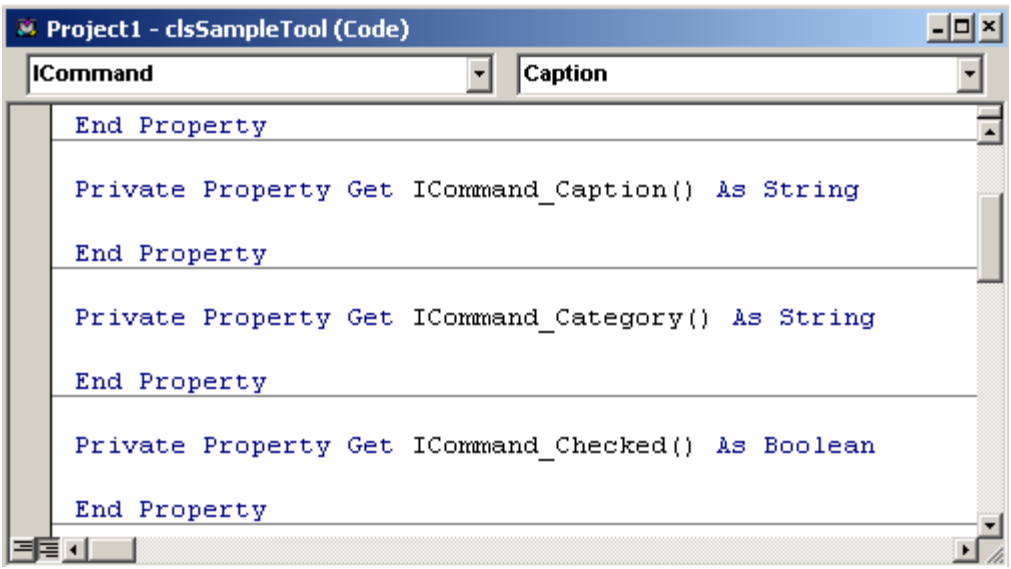

Figure 4.5 ICommand Stub Code

If a class implements an interface, it must implement all of that interface's properties and methods, even if the class does not provide code for some of those

properties and methods. Properties and methods are implemented by generating stub code as shown above. For this discussion, stub code is generated for the Bitmap, Checked, HelpContextID, HelpFile, and Message properties, but code to respond to calls to those properties is not written. Code for the other properties is written as shown below.

Property	Code
Caption	ICommand_Caption = "Show OIDs"
Category	ICommand_Category = "Sample Tools"
Enabled	ICommand_Enabled = True
Name	ICommand_Name = "SampleTools_ShowOIDs"
Tooltip	ICommand_Tooltip = "Display ObjectIDs for Selected Features"

Table 4.1 Implementation Code for ICommand Properties

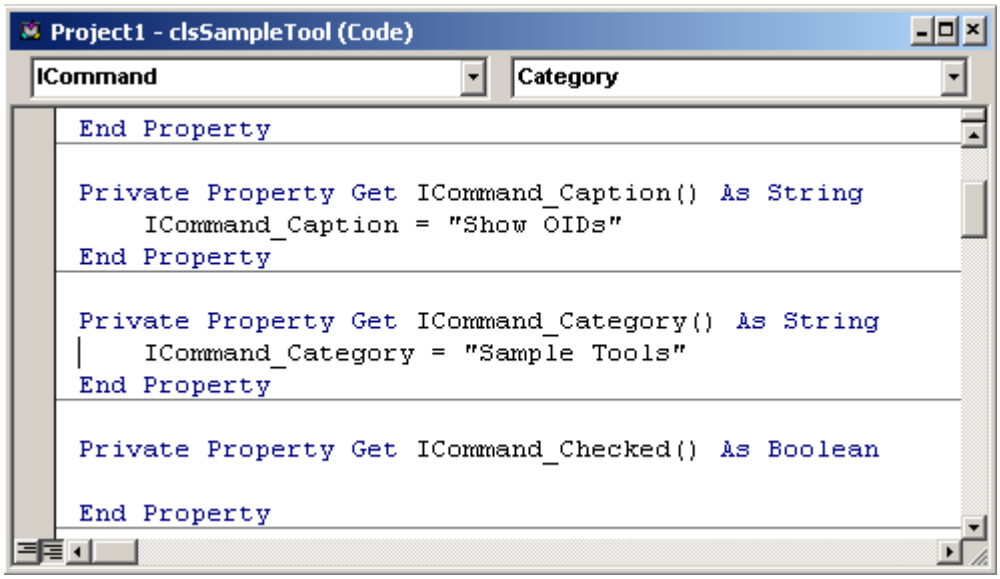

Figure 4.6 Implementation Code for ICommand Properties

The OnCreate procedure from ICommand is used to create a hook to the application that calls the DLL. In this example, the hook is a pointer to the ArcMap application that contains the custom tool. Without the hook, the tool would not know which ArcMap application it operates on. A module-level variable that stores an ArcMap Application object is set to the hook argument in the OnCreate procedure with the following code.

```
Set m_pApp = hook
```

Normally, this variable is stored as a global variable, g_pApp, and is declared in a separate module called modGlobals. However, for this example, all required functionality for the tool is coded in clsSampleTool (to keep things simple.) The purpose of modGlobals as well as other modules and forms that could be added to the project is discussed below.

4.1.1.2 Additional Modules and Forms

The class file that creates a custom tool may be supported by other modules and forms in the Visual Basic Project. In general, the tools created for this research included three modules (global variables, general utilities, network utilities) and three forms (input form, progress form, help form.) Modules differ from forms in that modules contain only code, while forms contain both code and a graphical component that interacts with the user.

Each of the modules is stored on disk. These modules do not change from project to project, except when new functions, subroutines, or declarations are added to them. Because the existing content of the modules is left unchanged, older projects that rely on the modules are not affected by these additions.

modGlobals is a module that stores global variables for use throughout the project. A global variable can be accessed by any component of the project. Global variables are useful for passing arguments, such as user inputs, between forms and classes. A global variable, g_pApp, is also used to store the hook to the application that calls the DLL.

modUtilsGeneral is a module that contains public functions and subroutines that are useful for many general applications. An example of a function from this module is GetFeatureLayer. This function returns a Feature Layer from a map document given the Feature Layer's name, which might be provided by the user on an input form.

modNetworkUtils is a module that contains a variety of network related functions and subroutines. Components in the project can make calls to this module to initialize a trace solver, create flags at points on the network, and perform other useful network tasks.

The input form is based on a template that already contains many of the elements and code required to process a user's inputs. When a new tool is created, the template form is copied and modified to suit the specific needs of the tool. Using a template form takes advantage of the reusable nature of input forms and saves much time in the development process of new tools. The template form contains four combo boxes (see Fig. 4.7). The first box and third box store the first layer and second layer that the user is interested in, respectively. The second box and fourth box store a field from the first layer and a field from the second layer that the user is interested in, respectively. Each combo box lists only valid

choices dependent upon the purpose of the tool. For instance, if a tool operates on polygon layers, then the combo box will not list the names of any point or polyline layers. OK, Cancel, and Help buttons are arranged at the bottom of the tool. The OK button places the user's choices into global variables so that those choices can be accessed by the class file that builds the tool. The Cancel button stops the operation. The Help button displays a help form that provides information about the tool.

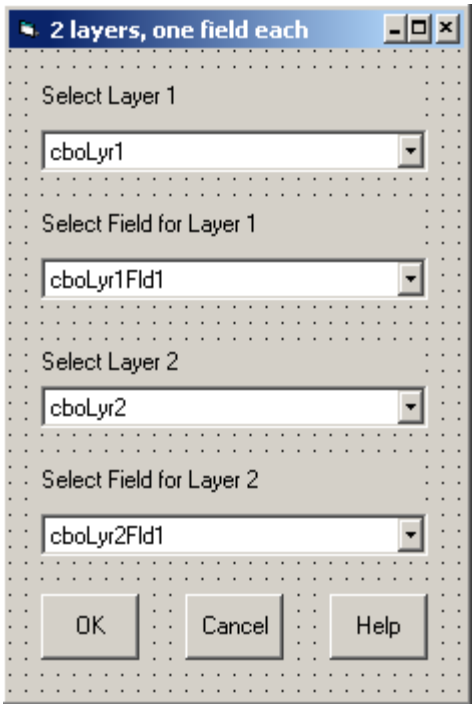

Figure 4.7 Template Input Form

The progress form (see Fig. 4.8) is a standard form stored on disk that can be added to any project, as long as Microsoft Common Controls 6 (MCC6) is installed on the computer. The form uses the progress bar control from MCC6 to

display the progress of a tool's operations. The form possesses a Cancel button that the user may click at any time to cancel the current task. Each tool created for this research is designed to gracefully recover from a cancel request without loss of data or crashing. The progress bar's properties can be accessed by other components in the project to set the status of the progress bar, so no modification is ever required to the progress form stored on disk. Because clsSampleTool does not perform lengthy operations, the progress form is not added to the project in this example.

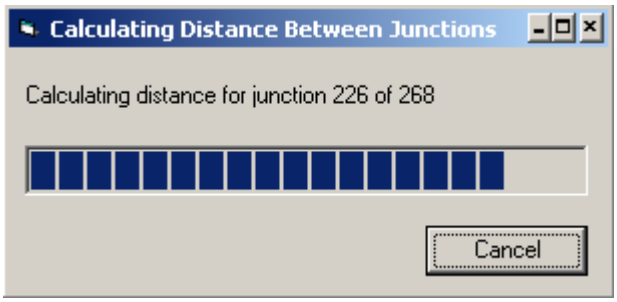

Figure 4.8 Sample Progress Form

The help form is a simple form that consists of a help message and an OK button. This form is created for each tool.

A Project Explorer window showing the modules and forms discussed above is shown in Fig. 4.9 for an example tool. Each component of the project is included in the final DLL that is produced.

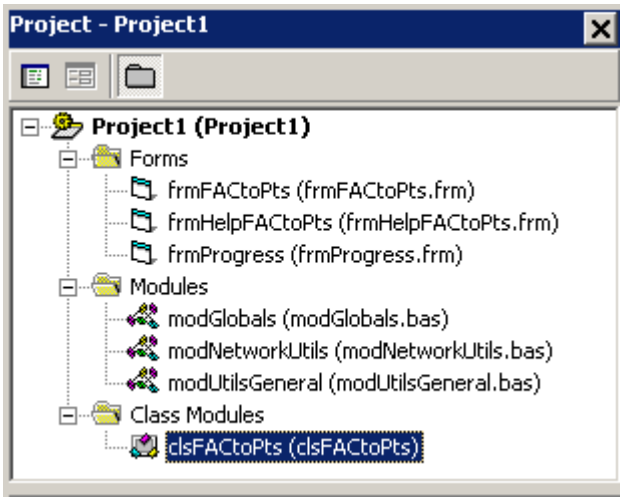

Figure 4.9 Module, Form, and Class Files in Sample Project

4.1.1.3 Code for the OnClick Procedure

clsSampleTool will be added to ArcMap as a command button. When the user presses the button, the OnClick procedure from the ICommand interface is called. Fill in the OnClick procedure in clsSampleTool with the code as shown below.

```
Private Sub ICommand_OnClick()
On Error GoTo errorhandler
    Dim Counter As Long 'universal counter
    Dim pMap As IMap
    Dim pDoc As IMxDocument
    Set pDoc = m_pApp.Document
    Set pMap = pDoc.FocusMap

    'Get the first layer in the map document
    Dim pFLayer As IFeatureLayer
    Set pFLayer = pMap.Layer(0)

    'Get the selected features
    Dim pFeatureSelection As IFeatureSelection
    Dim pSelectionSet As ISelectionSet
    Set pFeatureSelection = pFLayer
```

```
    Set pSelectionSet = pFeatureSelection.SelectionSet
    If pSelectionSet.Count = 0 Then
        MsgBox "Please select features first"
    Else
        Dim pFCursor As IFeatureCursor
        Dim pFeature As IFeature
        Set pFCursor = pFLayer.Search(Nothing, False)
        Set pFeature = pFCursor.NextFeature
        Do Until pFeature Is Nothing
            MsgBox "ObjectID = " & CStr(pFeat.OID), _
                , _
                "Sample Tool"
            Set pFeature = pFCursor.NextFeature
        Loop
    End If

    Exit Sub

errorhandler:
    MsgBox Err.Description, , "Error Number " & Err.Number
End Sub
```

This code gets a reference to ArcMap through the Application variable, m_pApp. It then gets a reference to the first layer in the map, which has an index of zero. The second layer has an index of 1, and so on. Next, the code creates a cursor, called pFCursor, which points to the selected Features in the layer. If no Features are selected, the code displays a message box asking the user to select features first. Otherwise, the cursor cycles through each selected Feature, displaying ObjectIDs in a message box. If any errors occur, a message box appears giving the error number and a description of the error.

4.1.1.4 The Importance of Error Handling

If a tool produces an error and does not adequately handle that error, then ArcMap considers the tool to be “broken” and will not allow further calls to the tool. If a tool is broken, then it will not function again until the tool’s DLL is

reinstalled on the computer (preferably with added error handlers.) For this research, errors were addressed in two ways: error prevention and error handling.

A useful method to prevent errors is to limit the range of inputs allowed by the user. For instance, if a tool adds values from a user-specified field to produce a total value, then the input form should limit the choice of fields to those that store numeric values. Similarly, tools that perform network operations should only display network layers as choices on the input form. Limiting user inputs to feasible values can save many hours of error handling work later on. Limiting user selections also benefits the user by removing many of the fields or layers that the user would not select anyway.

A second method for preventing errors is to check the nature of each value before processing that value. For instance, before an operation is performed on a value in a field, the tool should check to see if that value exists. Otherwise, the tool may attempt to perform an operation on a null value, resulting in an error. While this method is a more secure way of preventing errors than the previous method, it can also add an enormous amount of code to the project. A combination of both methods was found to be the best approach to error prevention.

If errors do occur during a tool's operation, a message box appears displaying a description and number for the error. In some cases, the location of the error within the code is also specified. While this technique may not be the best strategy for handling errors, it is easy to implement and satisfies ArcMap sufficiently so that the tool is not considered as broken.

4.1.1.5 Making the DLL

With code written for each component of the project, the next step is to make the DLL. From the Project menu, choose Project1 Properties. In the Project Properties window that appears, change the name of the project to Sample_Tool. Then click OK.

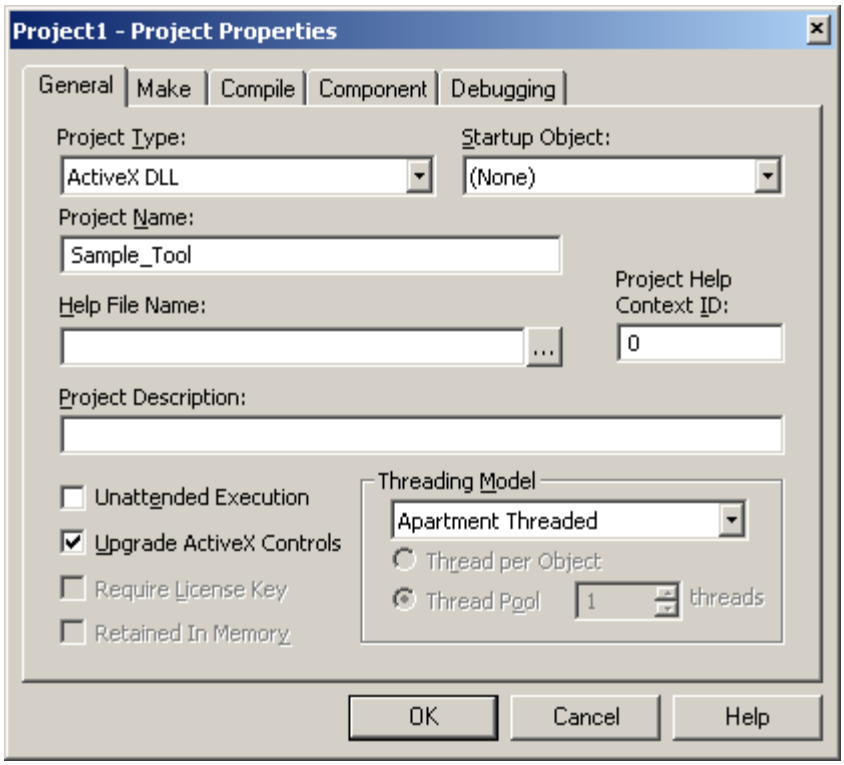

Figure 4.10 Project Properties Window

From the File menu, click Make Sample_Tool.dll. All of the code is compiled and checked for obvious errors. If no errors are found, Sample_Tool.dll is created in the specified directory.

4.1.2 Adding the DLL to ArcMap

Once the DLL is created, it can be added to ArcMap as a custom tool. Start ArcMap by clicking Start>Programs>ArcGIS>ArcMap. In the Tools menu, click Customize. In the Customize window, click the Commands tab. This window displays a list of all command buttons that can be added to the ArcMap document. To add the custom tool to this list, click Add From File. Navigate to Sample_Tool.dll and click Open. A window appears displaying which objects were added. Click OK to close the window. The Show OIDs tool is now displayed in the Customize window under the Sample Tools category.

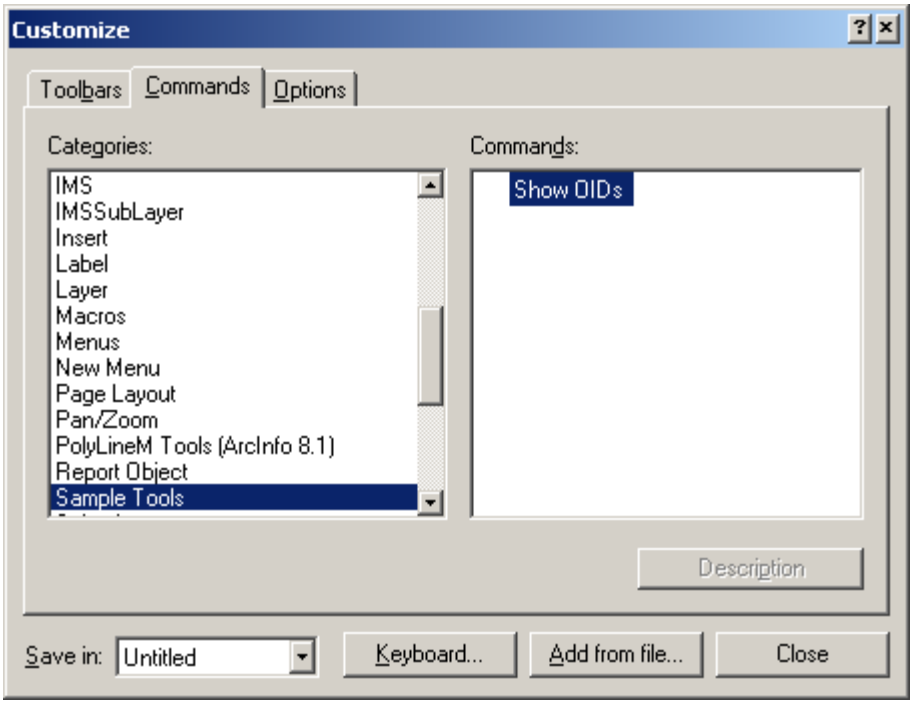

Figure 4.11 Show OIDs Tool in Customize Window

Drag the Show OIDs tool next to another button on the ArcMap interface. The tool is now ready for use.

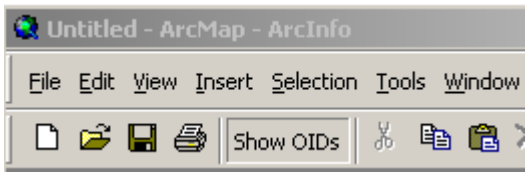

Figure 4.12 Show OIDs Button

If a layer is added to the document and Features are selected from that layer, a message box similar to the following will be displayed when the Show OIDs button is clicked.

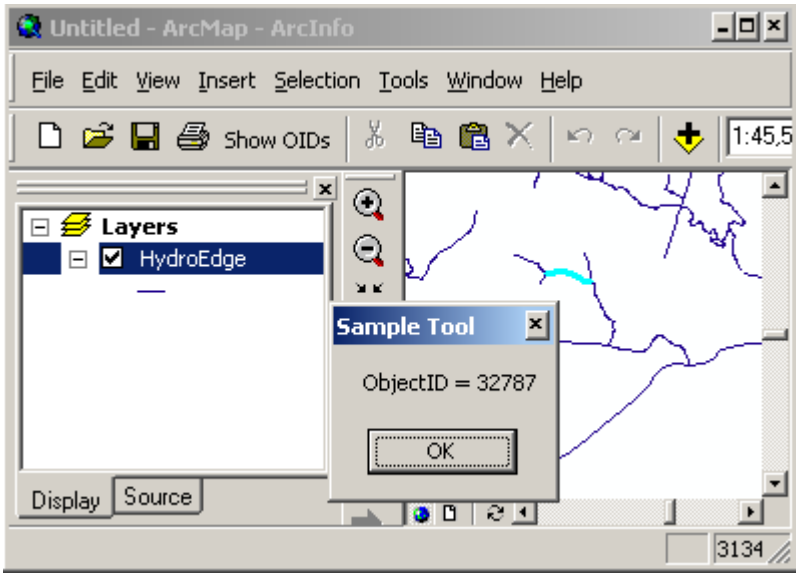

Figure 4.13 Sample Output from Show OIDs Tool

If no layers were present in the map, or if Features in the current layer had no ObjectID, then a message box would appear displaying an error.

4.2 APPLYING DESIGN PRINCIPLES

The design principles behind intelligent software construction and graphical user interface design were used as guidelines in the development process of the prototype Arc Hydro tools developed for this research. The general adherence to the guidelines is discussed below.

4.2.1 Software Construction

Although the tools created represent a relatively simple and straightforward software system, the principles of intelligent software construction were still applied to their development. This helps to insure that the tools will be reusable, and it also tests the extent that tools created for use in ArcGIS can adhere to the guidelines.

4.2.1.1 Robustness

Although originally intended to work with the structure and Feature Classes of the ArcGIS Hydro Data Model, each tool developed in this research has been designed to support operations on any Feature Class in the GIS, provided that the Feature Class contains the correct structure for each tool's operation.

If an error occurs during a tool's operation, and the tool does not adequately handle that error, ArcMap considers the tool to be broken and no longer allows access to the tool. To prevent this from occurring, each tool possesses a simple error handler and cleanup routine. The error handler displays a message box giving a description of the error, the error number, and in some cases the location in the code where the error occurred. The cleanup routine usually consists of hiding all forms associated with the tool, stopping any edit operations,

and cleaning up temporary objects from the computer's memory. While a more advanced error handling method could be applied, the message box method is easy to implement and prevents the tool from being considered broken by ArcMap.

Each tool requires ArcMap to be in an Edit session before the tool will proceed. By operating from within an edit session, the user has the option of ending the edit session without saving any changes that the tool made to the data. This proves valuable when an error occurs midway through a task, and only half of the data is processed. In this situation, the user has the option of fixing the error, or ending the edit session without saving edits to recover the old data.

4.2.1.2 Extensibility

As each tool was developed, repetitive routines from the tool's operation were separated into functions and subroutines outside of the tool's main procedure. Depending on how specific the routines were to the tool's purpose, they were either placed as an additional routine in the tool's main class module, or as a routine in a global utility module (such as modNetworkUtils.) As each routine was created, the code in the tool's main procedure became cleaner, more compact, and easier to read. This process also partitioned the main procedure into sections marked by calls to each routine. These factors improved the extensibility of the tool, since a modification to the tool's design could be isolated to a single partition.

The nature in which a tool's main procedure communicates with the form that accepts user inputs also promotes an extensible design. At first, the input

form writes the user's inputs to public variables declared in the form. The main procedure, usually located in the tool's class file, then reads from those public variables. However, as more tools were developed for a single DLL, this strategy lead to a large set of public variables scattered through the project. As a second approach, user inputs were stored in public variables in a single module. Still, there were a large number of public variables declared, even though they were all located in one place. This also led to confusion as to which variable belonged to which tool in the project. The current approach involves writing user inputs to a public dynamic array in a module. The main procedure then reads its required inputs from each item in the array. Now a single array can store the inputs from any number of user forms associated with any number of tools. More importantly, the main procedure of each tool does not care where the inputs came from, as long as they have been assigned to the public array. In this situation, the module with the public array acts as a switchboard between input and operation. This allows each tool to be extended to allow inputs from a variety of sources.

4.2.1.3 Reusability

Separating repetitive tasks into routines as described above also promotes a very reusable design. Because common routines are stored as public functions or subroutines in modules, they can be used by any tool in the project. The design of each tool is highly modular in general. Separate components control accepting user inputs, displaying status and progress of an operation, and performing the main operation required by a tool.

A direct application of reusability is the use of a template form to create all inputs forms for the tools. As early input forms were created, similarities were noted in each form's display, as well as the coding and functionality behind each form. This basic functionality was programmed into a template form that can be copied and modified as needed. The template form contains basic routines for accepting layer names and field names from the user, as well as routines for populating various components on the form with the appropriate values. Using the template form greatly reduces the development time for a given tool, since creating an effective user-interface is often be one of the most challenging aspects of the design process.

4.2.1.4 Compatibility

Because the tools were developed in a COM-compliant environment, they can easily include the functionality of other COM-compliant software. However, because the tools are designed to operate on ArcGIS objects, they have little potential to be included as components of an ArcGIS independent software system.

4.2.2 User Interface Design

The principles of good user interface design were useful in developing effective interfaces for the tools. Design aspects related to each of the seven principles are discussed below.

4.2.2.1 User in Control

All tools but two (Make Schematic Lines and Accumulate Area Values to Points) can run on all records in the feature class, or records selected by the user.

Each form that accepts user inputs, automatically screens out values that would obviously produce errors. However, the forms do allow users to select input values that may potentially cause problems (such as writing numeric string values to an integer field). In such situations, the form alerts the user to the problem that may arise, but still allows the user to proceed if so desired. The goal was to give the user a sense of being in control while minimizing the occurrence of user-generated errors.

Three tools (Store Area Outlets, Accumulate Area Values to Points, Find Distance Between Points) allow the user to cancel processing at any time during the tool's operation, without adverse effects on the stability of the application or the integrity of the data being manipulated. The cancel button is displayed on a progress form that appears during long operations.

4.2.2.2 Directness

The interface for each tool is relatively simple, with inputs at the top of the form and OK-Cancel-Help buttons at the bottom. OK, Cancel, and Help buttons are commonly used in other Windows applications, and their use is intuitive in each of the tools developed for this research. The format used by the tools for requesting input from the user strongly resembles that of the basic tools provided with ArcGIS software, with layer names given first, followed by field names or other parameters.

4.2.2.3 Consistency

Each tool uses the same format for the user interface, with the inputs at the top of the form and the OK-Cancel-Help buttons at the bottom of the form. Each

form is developed from the same template, to simplify the task of producing a consistent GUI design. The basic functionality required by input forms is already programmed into the template form, resulting in consistency both in the display of the form as well as the coding behind the form.

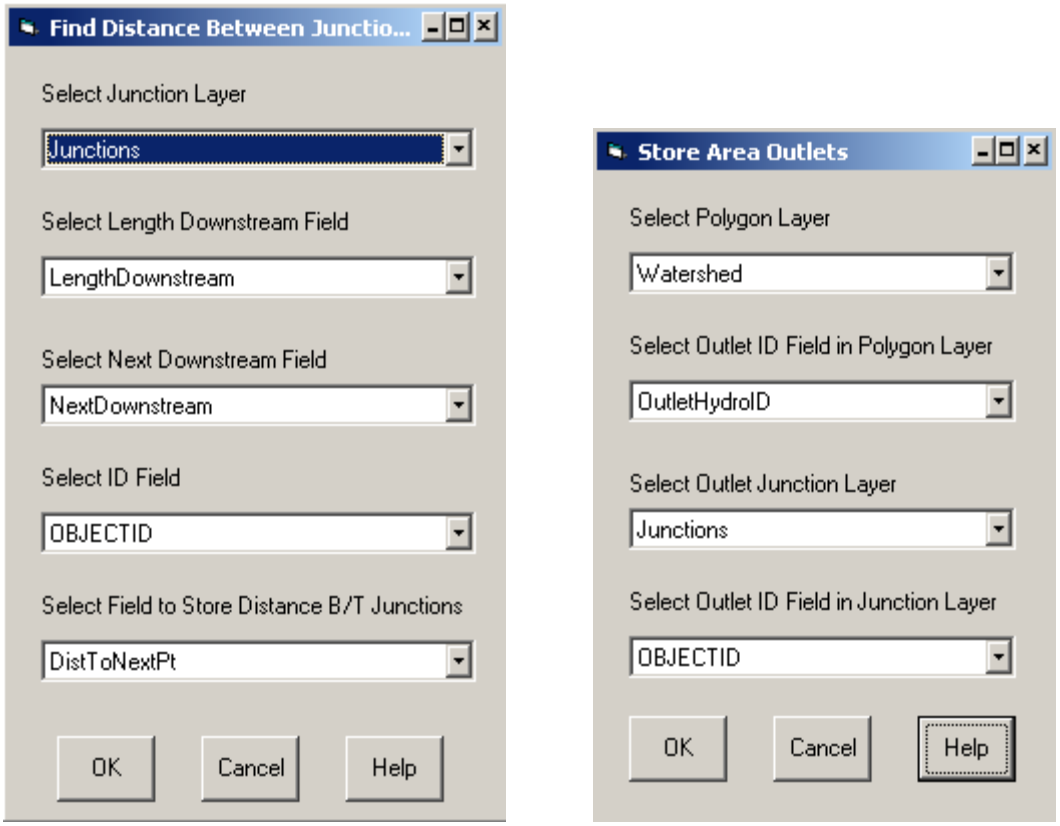

Figure 4.14 Comparison Between Form Layouts

In some cases, the course of action that a tool takes may vary depending on the user's inputs. When this occurs, the tool's operation appears the same to the user no matter which course of action the tool takes (when appropriate.)

4.2.2.4 Forgiveness

The tools possess error handlers to trap errors that occur during operations. In most cases, a message box appears on the screen describing the nature of the error, and sometimes giving the name of the component that generated the error. This type of "forgiveness" is extremely important within an ArcMap application. If a tool does not adequately handle errors, ArcMap considers the tool to be broken and allows no further access to the tool.

When prompting for user inputs, all forms automatically screen out values that are obviously incorrect, allowing the user to only select from a list of potentially acceptable choices. However, some freedom is still allowed for options that may or may not produce an error. In such situations, a message box appears alerting the user to the potential problem before the operation takes place. If the user decides to continue, and an error occurs, the error is caught by the tool's error handler and processed in a safe manner.

4.2.2.5 Feedback

All tools that perform tasks more than a few seconds long display a progress bar indicating the progress of the operation and the nature of the current operation. When a tool produces tangible output (a file saved to disk) that can be displayed in ArcMap, that output is automatically added to ArcMap to indicate success upon completion of the operation. When a tool changes the state of a component on the map display, the display is refreshed to indicate the changed state of the component. For example, if the flow direction for a particular edge is reversed, an arrow indicating the new flow direction is drawn on the display.

4.2.2.6 Aesthetics

When a tool requires a Feature Layer and fields from the Feature Layer, the user is prompted for the name of the Feature Layer first (at the top of the form), with the input boxes for fields aligned below the Feature Layer input box. The forms posses no fancy graphics or other items that may distract the user from purpose of the form. All items on a form are aligned properly, with minimal "dead space" and an intuitive arrangement of form elements.

4.2.2.7 Simplicity

A descriptive label is placed above each input box to indicate the purpose of the box. All input forms and help forms posses a caption that indicates their function. Input boxes are grouped according to the Feature Layer that they are related to. In developing these tools, following simplicity of design not only improved the quality of the user interface, but also decreased the time required to make modifications to the interface.

Chapter 5: Results

Eleven custom tools were developed for this research. These tools make up a prototype toolset that operates on the ArcGIS Hydro data model. The tools are designed to work in ArcMap as custom add-ins. Ten of these tools are located in an Arc Hydro Tools toolbar, while the remaining tool, an application designed to work with the USGS National Water Information System (NWIS), resides as a separate command button. The DLL and all source files used to create the Arc Hydro Tools and the NWIS application can be found in a CD at the back of this thesis.

These tools facilitate the integration of Arc Hydro with hydrologic simulation models through data preparation and by establishing connectivity between features in the landscape (through the use of key attributes). This connectivity information is essential to hydrologic simulation models, as these models pass time series information, such as rainfall/runoff data, between appropriate hydrologic features, such as a watershed and a monitoring gage located at its outlet.

5.1 ARC HYDRO TOOLS

The term Arc Hydro Tools refers to a set of 10 custom tools created in the Visual Basic programming environment to automate useful tasks within an ArcMap application. These tools are saved as an Active-X DLL (ArcHydroTools.dll) and can be added to an ArcMap document through the customization window in ArcMap. The tools represent an early attempt to

incorporate additional analysis and computational capabilities relevant to water resources applications into the ArcMap environment. As each tool was created, an improved knowledge of general programming techniques and specific strategies to work with ArcObjects was acquired. As a result, more recent tools in the toolset exhibit a more efficient and intelligent programming style, with a more modular structure, a more robust design, and better potential for reusability. The more recent tools also reflect a smoother integration with the ArcMap user interface, incorporating functionality such as a cancel button for long tasks.

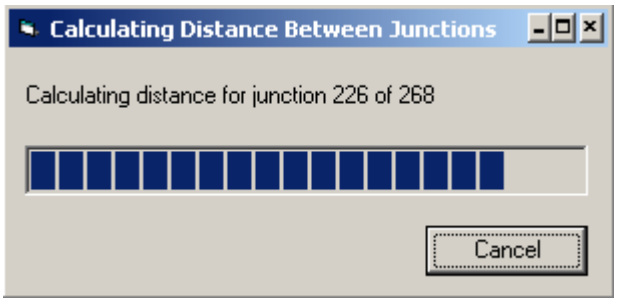

Figure 5.1 Sample Progress Form with Cancel Button

5.1.1 Arc Hydro Tools Toolbar

The Arc Hydro Tools toolbar contains all of the Arc Hydro Tools organized into toolbar items and menus. Once installed, the Arc Hydro Tools toolbar can be added to an ArcMap document like any other ArcMap toolbar.

5.1.1.1 Description

The Arc Hydro Tools toolbar contains three main items: An Arc Hydro Tools menu, a Point Nav menu, and Make Schematic button. The Arc Hydro Tools menu contains tools for performing various tasks in populating the

attributes of the ArcGIS Hydro data model. This menu contains three items: an Assign HydroID button, a Downstream Length menu, and a Flow Direction menu. The Downstream Length menu has two items: a Calculate for Edges button and an Assign to Junctions button. The Flow Direction menu has two items: a Store Flow Direction button and an Assign Flow Direction button.

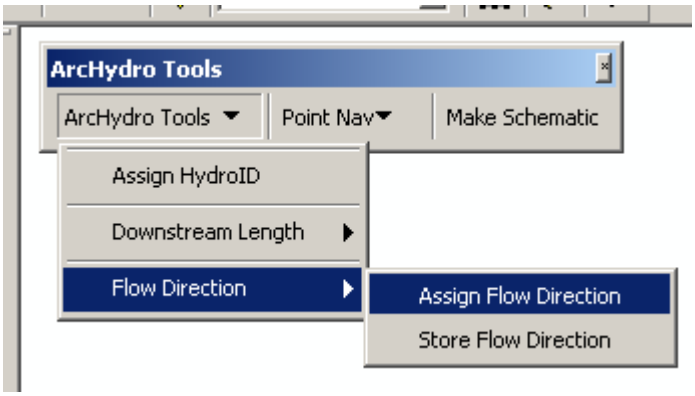

Figure 5.2 Arc Hydro Tools Toolbar

The Point Nav menu contains tools for developing relationships between points in a network and accumulating values through them in the downstream direction. The Point Nav menu contains four items: a Find Next Downstream Button, a Store Area Outlets button, an Accumulate Areas to Points button, and a Find Distance Between Junctions button.

Each of the buttons on the toolbar can be added individually to the ArcMap user interface through the customization window. As with all ArcMap command buttons, they can also be grouped into different toolbars with generic ArcMap buttons if so desired. A table providing a brief description of the purpose

of each tool is shown below. Each of the buttons in Arc Hydro Tools is described in the remainder of this chapter.

Tool	Purpose
Assign HydroID	Assign HydroIDs to features on the map
Calculate Downstream Length	Calculate downstream length for edges
Copy Downstream Length to Junctions	Copy downstream length to junctions
Assign Flow Direction	Assign flow direction to edges
Store Flow Direction	Store flow direction of edges in table
Find Next Downstream	Find next downstream junction
Store Area Outlets	Find outlets for areas
Accumulate Area Values to Points	Accumulate values from areas to points
Find Distance Between Junctions	Find distance between junctions on a network
Make Schematic Lines	Make a schematic lines from a set of points

Table 5.1 Function of Arc Hydro Tools

5.1.2 Assign Hydro ID

This tool assigns HydroIDs to Feature Classes and tables in the current map that have a HydroID field of type String. This tool is designed to operate on the HydroID attribute that is present in all Feature Classes in the ArcGIS Hydro data model.

5.1.2.1 Description

A HydroID is a unique identifier across a geodatabase. The ID is built from the object class ID of the table or Feature Class and the ObjectID of each row or feature in the table or Feature Class. The object class ID is a unique identifier for a Feature Class or table in a geodatabase. The ObjectID is a unique identifier for a row or feature in a table or Feature Class. The ObjectID starts at 1 and increments by 1 with each new row added to the table. Similarly, the object

class ID starts at 1 and increments by 1 with each new class created in the geodatabase. For instance, if a geodatabase contains three Feature Classes, their object class IDs may be 1,2, and 3, respectively.

The HydroID is built by concatenating the object class ID and the ObjectID into a single ID. Because ObjectIDs are unique in a Feature Class or table, and because object class IDs are unique in a geodatabase, the combination of object class ID and ObjectID is unique across a geodatabase. The HydroID has the following format: CC000000, where "CC" represents the object class ID, and "000000" represents the ObjectID. The format does not support geodatabases with more than 99 Feature Classes and tables. It also does not support Feature Classes or tables with more than 999,999 records. However, such situations are highly unlikely to occur, as such a geodatabase would be too cumbersome to be practical.

As an example of how a HydroID is created, consider a HydroEdge feature in the HydroEdge Feature Class. The third HydroEdge (with ObjectID = 3) in the Feature Class (with object class ID = 2) would have a HydroID of 02000003.

The Assign HydroID tool assigns HydroIDs to rows that have a HydroID field of type String. If the HydroID field is not found in the table, the tool skips that table and moves on to the next one. Note that those Feature Classes and tables that are in the database, but not registered with the geodatabase always have an object class ID of -1. If more than one such table or Feature Class exists in a geodatabase, then the HydroID may not be unique across the geodatabase.

The tool can run on a selected set of records or all records. If no features are selected, the tool runs on all records. If any features are selected, the tool runs only on the selected records.

5.1.2.2 Beneficial Uses

A unique HydroID identifier is useful for a variety of reasons. In a hydrologic context, a unique ID allows each feature to be treated as an individual component in a hydrologic system. Each feature can be "connected" to other individual features from any Feature Class by storing the HydroID of one class as an attribute of another. This is similar to the manner in which objects in HEC-HMS are connected (HEC, HEC-HMS). Thus a watershed can contain the HydroID of the junction it drains to, while a junction can contain the HydroID of the next downstream junction it flows to, and so on.

5.1.2.3 Limitations

At present, the tool does not allow the user to select which field to assign HydroIDs to. In order to assign HydroIDs, a field named "HydroID" must appear in the table. The field type for HydroID must also be of type String. There are two reasons for required a string field type: 1) HydroID is defined as a string type in the ArcGIS Hydro data model, 2) a string type supports a wider variety of HydroID formats than a numeric type.

The tool also does not allow the user to choose the format for the HydroID. There are a number of schemes for applying a unique identifier to features across a geodatabase. The Assign HydroID tool only uses one of them.

5.1.3 Calculate Downstream Length

This tool calculates the length from the most downstream node on each edge in a given layer to the sink that the edge flows to. It then populates the specified length downstream field with those calculated values. This tool is designed to operate on the LengthDownstream attribute of HydroEdge in the ArcGIS Hydro data model.

5.1.3.1 Description

The LengthDownstream value starts at zero at a sink and increases in the upstream direction. The LengthDownstream value for a given edge includes the lengths of all downstream edges, but not the length of the current edge. Thus the length downstream for an edge at the most upstream segment of a river is the entire length of the river minus the length of that upstream segment. Likewise, the length downstream for any edge that is connected to a sink is zero. The tool works by tracing downstream from each edge in the network. The sum of length values from all edges returned in the trace (except for the current edge) is written to the length downstream field in the current edge.

Before using this tool, flow direction must be set in the network. If no flow direction is set, then the length downstream for every edge is zero (because the edge does not know where to flow.)

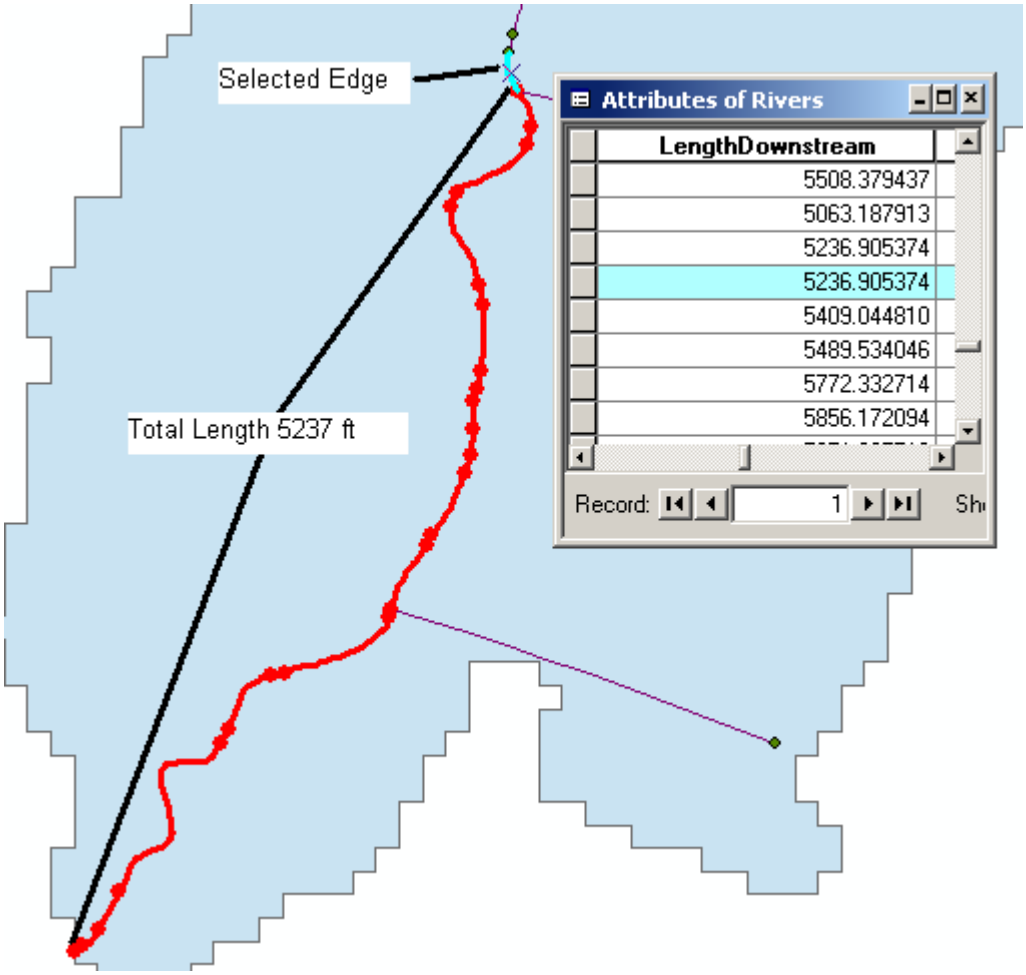

Figure 5.3 Result of Downstream Length Computation

The tool can run on a selected set of records or all records. If no features are selected, the tool runs on all records. If any features are selected, the tool runs only on the selected records.

5.1.3.2 Beneficial Uses

LengthDownstream may be used to populate measure values on edges. LengthDownstream may also be used in conjunction with flow velocity to

compute travel times in rivers. In hydrologic computational algorithms, LengthDownstream may be used to determine which features are processed first (those with the greatest LengthDownstream) in a system where backwater effects are negligible.

5.1.3.3 Limitations

The Calculate Length Downstream tool requires that a field to store length downstream already exists in the Feature Class. The tool should be run on a non-branching, non-looping network, or else incorrect downstream length values may be calculated. Non-branching means that the network does not branch in the downstream direction (as may occur with diversions.) Branching in the upstream direction is allowed.

The tool will not work correctly if there are complex edges in the network *and* some edges join other edges at anywhere but their endpoints. The tool works by adding up the lengths of all downstream features. If an edge joins a downstream edge in the middle of the downstream edge (as is possible with complex edges), the downstream edge's entire length is added to the upstream edge's downstream length total, when in fact only the portion of the downstream edge's length that should be added is the portion below where the upstream edge joins.

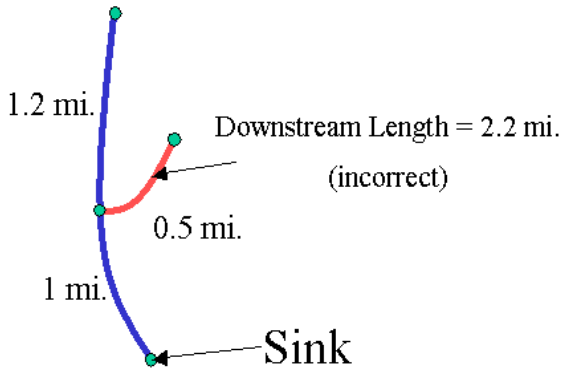

Figure 5.4 Incorrect Downstream Length Calculation for Complex Edges

To work properly in such situations, a measure system would have to be defined on the edges, and the tool would have to be revised to read values from the measure system.

This tool is designed to work with a network with a single Edge Feature Class. If the network has more than one Edge Feature Class, the tool may not work correctly.

5.1.4 Assign Downstream Length to Junctions

The tool reads LengthDownstream values from edges and writes those values to junctions in the network. This tool is designed to operate on the LengthDownstream attribute of HydroJunction in the ArcGIS Hydro data model.

5.1.4.1 Description

Once downstream length values have been calculated for edges, they may be copied to junctions using this tool. For each junction, the tool finds all edges connected to that junction, reads the downstream length values from those edges,

and then selects the edge with the smallest downstream length. In a non-branching (in downstream direction), non-looping network, this edge will be the only edge that the junction flows to, and it represents the most downstream edge that the junction is connected to. The tool adds the downstream length value of the edge to the edge's length to produce the downstream length value for the junction. If the downstream length values for all edges connected to a junction are zero, then this junction is treated as a sink, and it is assigned a downstream length value of zero. If no edges are found, a value of zero is assigned for the downstream length.

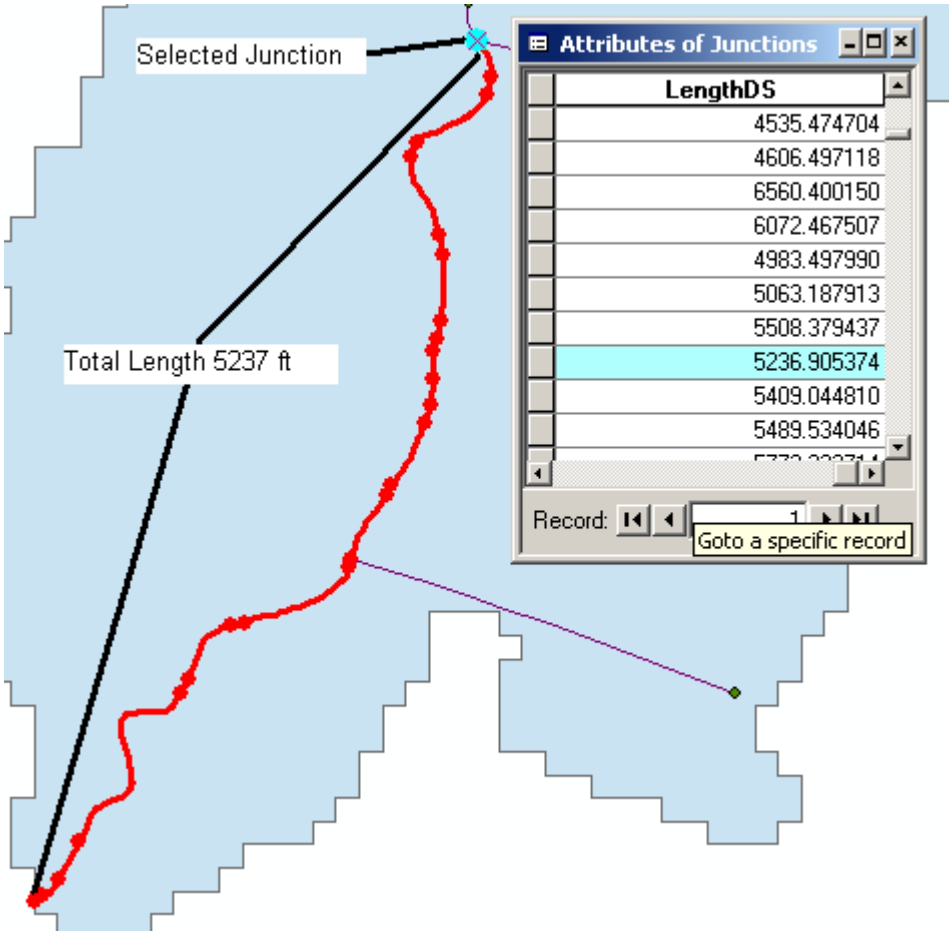

Figure 5.5 Result of Assign Downstream Length to Junctions Computation

The tool can run on a selected set of records or all records. If no features are selected, the tool runs on all records. If any features are selected, the tool runs only on the selected records.

5.1.4.2 Beneficial Uses

Calculating the difference between downstream length values for two points of interest gives the distance between those points along the network. In computational algorithms, downstream length may be used to determine which

features are processed first (those with the greatest downstream length) in a system where backwater effects are negligible. An example of such an algorithm is an accumulation routine where values from upstream junctions are passed down and added to values on downstream junctions.

5.1.4.3 Limitations

The Assign Downstream Length to Junctions tool requires that a field to store LengthDownstream already exists in the Feature Class. The tool should be run on a non-branching, non-looping network, or else incorrect downstream length values may be calculated.

In some networks, some junctions may be spatially coincident with each other. When this happens, the network builder chooses one and only one junction at a given location to be connected to the network. The other junctions just 'float' in the same location without being connected to other network features. This tool looks for this type of junction and copies the length downstream values from the junction that IS on the network to the other spatially coincident junctions. However, if the network includes junctions from another Feature Class that are spatially coincident with the junctions of interest, and if the junction from the other Feature Class happens to be the one connected to the network, then the tool may not work correctly and a value of zero for length downstream will be assigned to the junctions of interest at that location.

The tool will not work correctly if the junctions are on the interior of complex edges. If the tool selects the complex edge as the most downstream edge that the junction is connected to, it will add the entire shape length of the complex

edge to its downstream length value, instead of correctly adding only the portion of the edge that is downstream of the junction. To work properly in such situations, a measure system would have to be defined on the edges, and the tool would have to be revised to read values from the measure system.

This tool is designed to work with a network with one edge Feature Class. If the network has more than one edge Feature Class, the tool may not work correctly.

5.1.5 Assign Flow Direction

This tool assigns flow direction to edges in a network, either by choosing a flow direction to assign, or by reading values from a table. This tool is designed to operate on the FlowDirection attribute of HydroEdge in the ArcGIS Hydro data model.

5.1.5.1 Description

Flow direction in a network is stored as esriFlowDirection constants. These constants are shown in Table 5.2.

Constant	Value	Description
esriFDUninitialized	0	The flow direction is uninitialized.
esriFDWithFlow	1	The flow direction is in the direction of digitization.
esriFDAgainstFlow	2	The flow direction is opposite the direction of digitization.
esriFDIndeterminate	3	The flow direction is indeterminate.

Table 5.2 esriFlowDirection Constants

This tool accesses the flow direction for a set of edges using the network, and then assigns the flow direction for the edges. When assigning flow direction, the user may either choose one of the four esriFlowDirections to assign to all edges, or use values from a field in the edge Feature Class. The values in the field must correspond to esriFlowDirection constants, i.e. 0, 1, 2, or 3. If values in the flow direction field do not correspond to esriFlowDirection constants, the esriFlowDirection chosen by the user will default as the flow direction assigned to the edge.

The figures below show how the Assign Flow Direction tool can be used to set the flow direction of edges in the network. The first figure shows the default flow direction based on sinks in the network. The second figure shows the flow moving towards a different location after the tool was used.

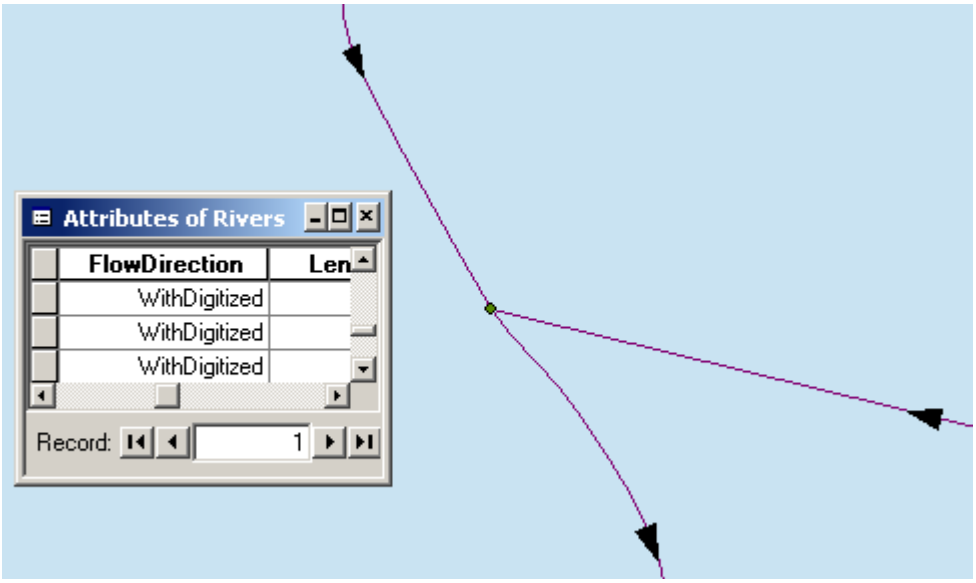

Figure 5.6 Default Flow Direction in a Network

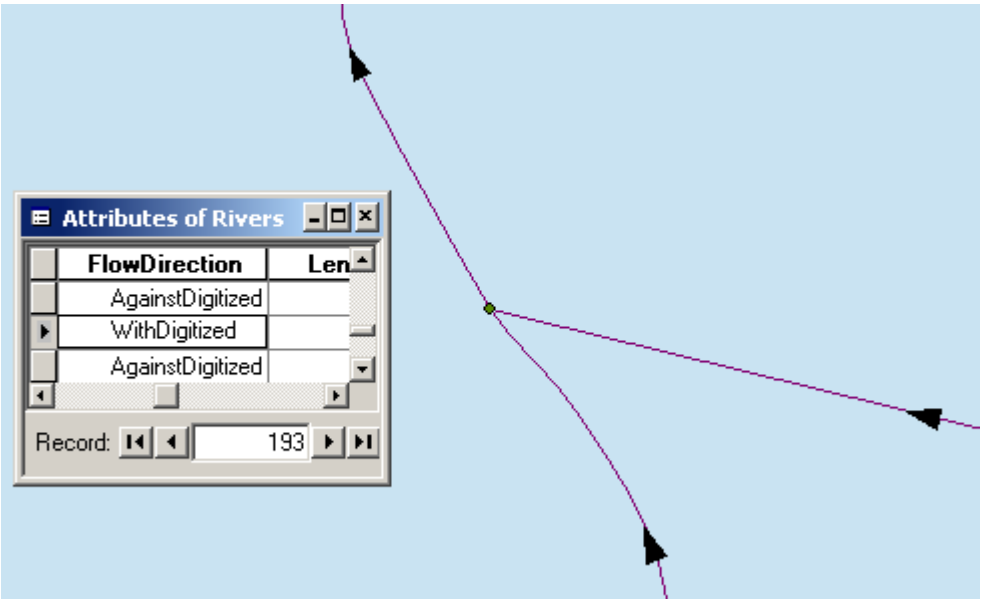

Figure 5.7 Result of Changing Flow Direction with Assign Flow Direction Tool

The tool can run on a selected set of records or all records. If no features are selected, the tool runs on all records. If any features are selected, the tool runs only on the selected records.

5.1.5.2 Beneficial Uses

Assigning flow direction based on an attribute allows the user to delete the network (for maintenance or distribution reasons) and still retain the proper flow direction values. Once the network is reestablished, flow direction can be assigned by attribute without having to create sinks in the network. Flow direction for indeterminate cases (such as edges in loops) can be assigned manually with this tool. Situations in which flow directions may change (such as in canals in flat areas) can also be modeled.

5.1.5.3 Limitations

A network must be present to assign flow direction.

5.1.6 Store Flow Direction

This tool reads the flow direction for a set of edges from the network and writes the value of the flow direction to the Feature Class. This tool is designed to operate on the FlowDirection attribute of HydroEdge in the ArcGIS Hydro data model.

5.1.5.1 Description

Flow direction in a network is stored as esriFlowDirection constants. These constants are shown in Table 5.2. The Store Flow Direction tool reads edge flow directions from the network and writes the esriFlowDirection values to the table of the edge Feature Class. The field chosen to store esriFlowDirection values must be of a numeric type.

The tool can run on a selected set of records or all records. If no features are selected, the tool runs on all records. If any features are selected, the tool runs only on the selected records.

5.1.6.2 Beneficial Uses

This tool allows manual editing of flow direction. Once flow direction is stored, the network may be deleted and flow directions will still be known on the edge (to the user, but not to the software). Also, storing flow direction makes possible the assignment of flow direction based on an attribute in a table.

5.1.6.3 Limitations

The field chosen to store flow direction must be of a numeric type. This helps to prevent users from inadvertently inputting invalid flow directions, such as text or dates.

5.1.7 Find Next Downstream

The Find Next Downstream tool uses the network to find the next downstream junction in a particular Feature Class, and assigns the ID of the downstream junction to a downstream junction ID field in the Feature Class table. This tool is designed to operate on the NextDownstream attribute of HydroJunction in the ArcGIS Hydro data model.

5.1.7.1 Description

For each junction in the network, this tool creates a flag at that junction and barriers at all other junctions. The tool then traces downstream from that junction and returns the ID of the junction stopping the trace. This is the ID of the next downstream junction. This ID is written to a field in the junction Feature Class table. The tool requires that flow direction is already set in the network.

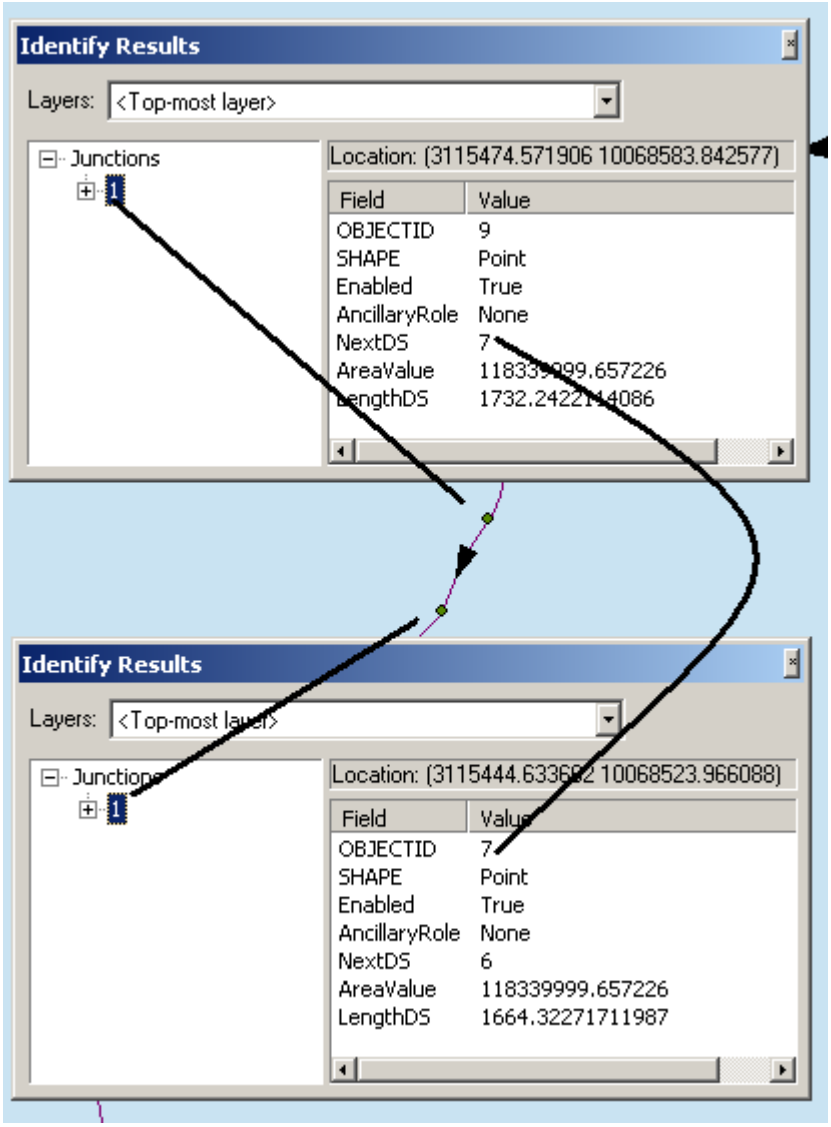

Figure 5.8 Results of Find Next Downstream Computation

The tool can run on a selected set of records or all records. If no features are selected, the tool runs on all records. If any features are selected, the tool runs only on the selected records.

5.1.7.2 Beneficial Uses

Having the next downstream junction listed as an attribute in the table allows junctions to "communicate" with each other without the presence of a network, passing values or other information as desired. Knowing the next downstream point of interest is also important for applications such as water rights analysis and Total Maximum Daily Load studies.

5.1.7.3 Limitations

Some junctions in the network may be spatially coincident with each other. When this happens, the network builder chooses one and only one junction at a given location to be connected to the network. The other junctions just 'float' in the same location without being connected to other network features. This tool looks for this type of junction and copies the downstream junction ID values from the junction that IS on the network to the other spatially coincident junctions. However, if more than one junction is in the next downstream location for a given upstream junction, then only one of the downstream junction's IDs (the one that is connected to the network) will be assigned to the upstream junction. If no downstream junction is found, the tool assigns a value of -1 to the downstream ID.

This tool will not work correctly if the junctions are on the interior of complex edges. It will also not work correctly if there are junctions from another Feature Class participating in the network that are spatially coincident with the junctions of interest, unless those junctions are sinks.

5.1.8 Store Area Outlets

This tool determines the outlet junction for an area and assigns the ID of that junction to a Junction ID field in the area Feature Class table. This tool is designed to operate on the JunctionID field of polygon feature classes in the ArcGIS Hydro data model.

5.1.8.1 Description

To perform hydrologic analyses involving rainfall/runoff and channel routing, areas must somehow be connected to the river network in order to pass runoff to the river channel. This tool is designed to facilitate a scheme where areas are connected to the network through outlet junctions. Each area in this scheme possesses an OutletID attribute. This attribute stores the ID of the junction that serves as the outlet for the area. Hypothetically, a junction may serve as the outlet of multiple areas, but each area will have only one outlet.

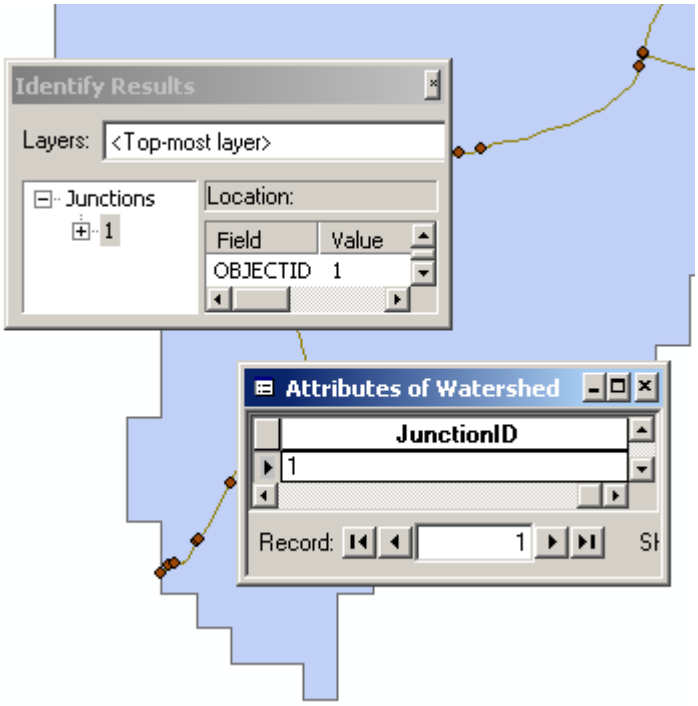

Figure 5.9 Results of Store Area Outlets Computation

This tool works by selecting all junctions along the boundary of an area. For each selected junction, the tool creates a flag at that junction and barriers at all other junctions along the boundary. The tool then traces upstream from the junction flag and counts the number of resulting edges that occur inside the area. If the number of edges is greater than zero, then this junction is a possible outlet junction. In most situations, only one junction will return edges inside the area after an upstream trace. However, situations may occur when there is more than one junction that returns a positive number of edges in the area after an upstream trace. When this happens, the junction with the most number of edges inside the area is designated as the outlet junction. The following figures illustrate the

process of selecting the outlet junction. The tool is operating on the pink area for this example.

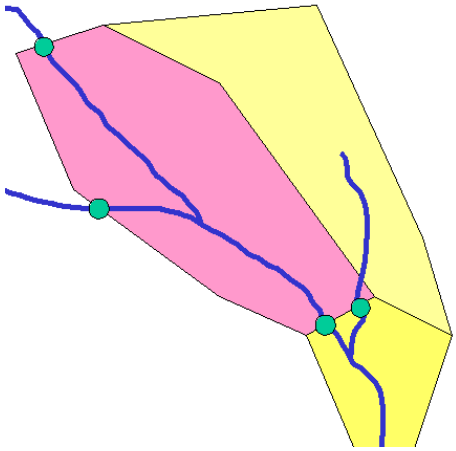

Figure 5.10 First Pass: All Junctions on Boundary

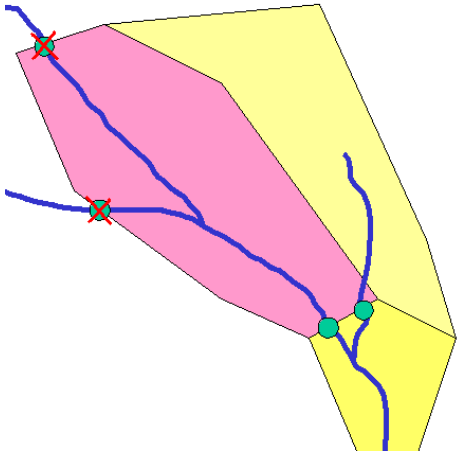

Figure 5.11 Second Pass: Junctions Producing Upstream Edges in Area

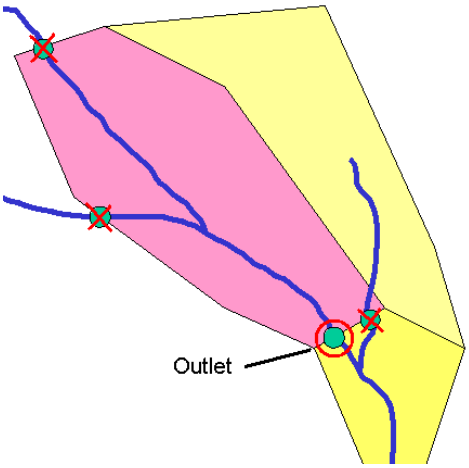

Figure 5.12 Third Pass: Junction With Most Upstream Edges in Area

If only one junction is found, a downstream trace is performed from that junction. If any resulting edges are found to be inside the polygon, then this junction is not designated as an outlet. Otherwise, it is designated as an outlet. If no junctions return edges in the area after an upstream trace, and there is more than one junction along the boundary of the area, then no junction is designated as the outlet.

If no outlet is assigned to an area, or if an area has more than one potential outlet according to the algorithm described above, then the ObjectIDs of those areas are copied to the Windows clipboard, and the user is notified. From there, they may be pasted into a document or spreadsheet so that the user may inspect each area manually.

The tool can run on a selected set of records or all records. If no features are selected, the tool runs on all records. If any features are selected, the tool runs only on the selected records.

5.1.8.2 Data Preparation

This tool requires some data preparation before it can be run. First, the polylines that represent the network should be intersected with the areas of interest to break the lines where they intersect the boundary of the areas. This may be done with the Geoprocessing wizard in ArcMap. The resulting line set should then be built into a network, producing a set of junctions. Some of these junctions will occur at the intersection of the areas and the lines. Outlet junctions will be selected from these junctions. A network must be loaded, and flow direction must be set on the network before running the tool.

5.1.8.3 Beneficial Uses

By connecting areas to the network through outlet junctions, areas can pass information to the network, such as runoff or pollutant loads. If the junctions also possess the NextDownstream attribute, then navigation of the landscape can be performed with the network as well as through areas.

5.1.8.4 Limitations

To insure that all junctions along the boundary of an area are selected, the tool builds a buffer polygon which is one map unit (e.g. meter or foot) bigger than the area, and another buffer polygon one map unit smaller than the area. The tool then selects all junctions between those buffer polygons. As a result, working in a coordinate system where 1 map unit corresponds to a great distance, such as a geographic coordinate system, may lead to more junctions being selected than intended. Data should be projected to a coordinate system with map units that

provide sufficient precision in measurements before running the Store Area Outlets tool.

5.1.9 Accumulate Area Values to Points

This tool accumulates values from areas to the outlet junction of the areas, and then accumulates those values through each downstream junction in the network. This tool is designed to operate on the JunctionID attribute of polygon Feature Classes, and the LengthDownstream and NextDownstream attributes of HydroJunction in the ArcGIS Hydro data model.

5.1.9.1 Description

Once areas have been connected to the network through outlet junctions, values can be passed from the areas to the network through the junctions. The Accumulate Areas to Points tool demonstrates a simple application of accumulating values from areas onto the outlets of those areas, and then passing the accumulated values downstream from junction to junction by using the NextDownstream ID on the junction Feature Class. Because this tool requires that the Junction ID of the areas, the NextDownstream ID on the junctions, and the LengthDownstream on the junctions be populated, no network is required to navigate from areas to junctions, and from one junction to another.

The tool starts by setting the field chosen to store the values in the junction Feature Class to zero. It then starts with the junction that has the greatest downstream length and finds all areas for which that junction is the outlet. It then adds all the values from those areas to the value field in the junction Feature Class table. Next, the tool finds the next downstream junction for the given junction

and adds its value to its downstream neighbor. The tool then moves on to the junction with the next greatest downstream length and iterates until all junctions have been processed.

By working from the greatest downstream length downward, the tool insures that values for all upstream areas are summed before processing downstream areas. This scheme is useful when no "backwater" effects occur in the area/point network.

The following figure shows the results of running the tool where the accumulated value was the area of each polygon in acres. The figure shows the outlet point for the most upstream area in the network.

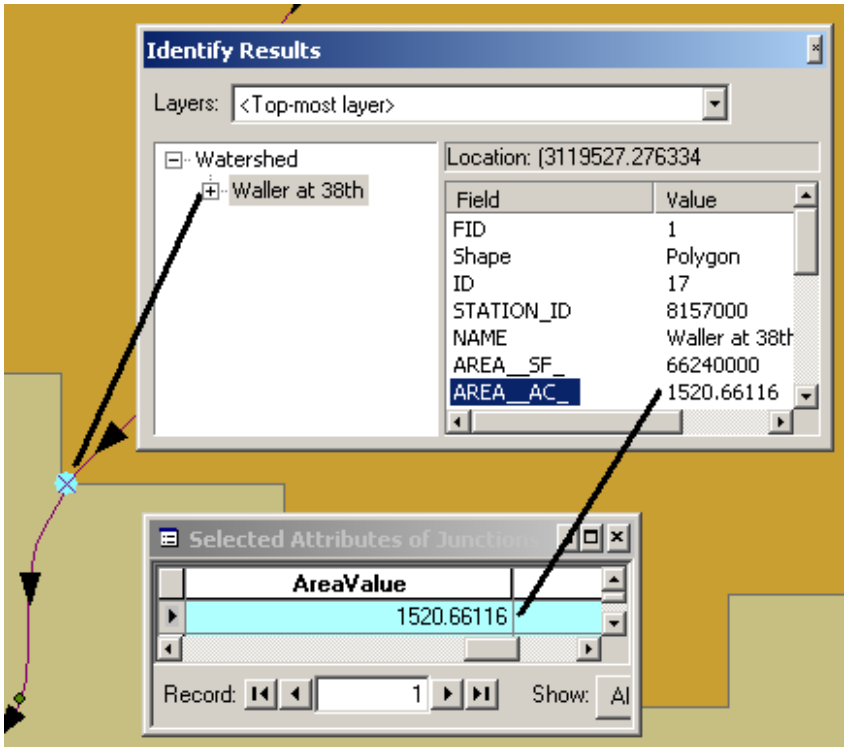

Figure 5.13 Results of Accumulate Areas to Points Computation

5.1.9.2 Beneficial Uses

This tool serves a number of uses relating to hydrologic computations. Any application which requires information to be passed from areas onto the network may benefit from this tool or some variation thereof. The algorithms used in connecting areas to points, and points to their downstream points, may prove useful in another application that requires navigation through the landscape.

5.1.9.3 Limitations

At present, the tool can only run on the entire junction Feature Class. Letting the user accumulate area values for a few, possibly disconnected junctions may produce misleading and inaccurate results. A future revision to the tool may involve allowing the user to select a set of junctions, and then accumulating values for ALL areas and junctions that flow to those selected junctions.

5.1.10 Find Distance Between Junctions

This tool finds the distance between a point or junction and its next downstream neighbor. This tool is designed to operate on the NextDownstream and LengthDownstream attributes of HydroJunction in the ArcGIS Hydro data model.

5.1.10.1 Description

If next downstream ID and length downstream have been populated for a set of junctions, finding the distance between a junction and its next downstream neighbor is a relatively simple task. This tool performs that task by subtracting the downstream lengths of two junctions to determine the distance between them.

If no next downstream junction is found, or if the length downstream values are corrupted, then the tool assigns a value of -1 as the distance between junctions.

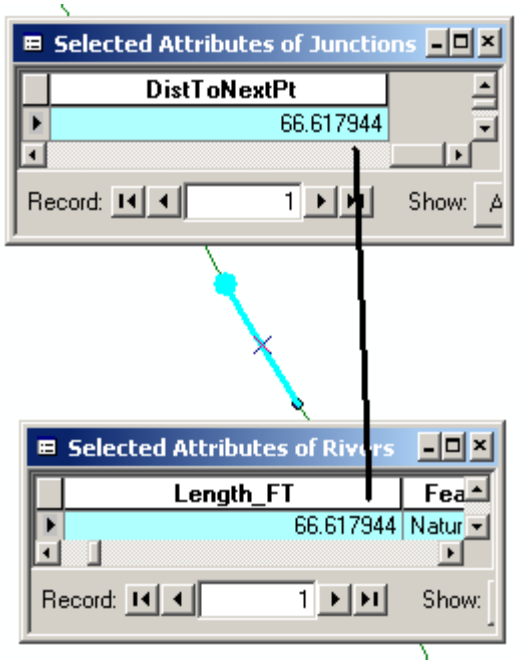

Figure 5.14 Results of Find Distance Between Junctions Computation

The tool can run on a selected set of records or all records. If no features are selected, the tool runs on all records. If any features are selected, the tool runs only on the selected records.

5.1.10.2 Beneficial Uses

Knowing the distance between points is useful in determining time of travel between points in a network. The information is also useful in water rights analysis and Total Maximum Daily Load analyses.

5.1.10.3 Limitations

The tool requires that the next downstream ID and length downstream attributes are already populated on the point or junction Feature Class. However, no network is required to run this tool once those attributes have been populated.

5.1.11 Make Schematic Lines

This tool creates a schematic line Feature Class from a set of points or junctions. This tool is designed to operate on the NextDownstream attribute of HydroJunction in the ArcGIS Hydro data model.

5.1.11.1 Description

If next downstream ID is known for a set of junctions, then a schematic line set can be creating connecting those junctions. A schematic line is a simple, straight line that directly connects two topologically related points in a network. Schematic networks are simplified versions of actual networks and are often used in hydrologic and hydraulic models to represent the hydrological system. The tool runs on the entire set of points or junctions from the given Feature Class.

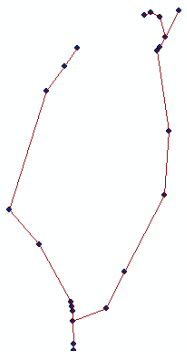

Figure 5.15 Results of Make Schematic Lines Operation

5.1.11.2 Beneficial Uses

A schematic line set is useful in creating a simple network that possesses all of the necessary information to perform a hydrologic or hydraulic analysis, without all of the cartographic information normally found in representations of river networks in Geographic Information Systems. A schematic network may give a clearer picture of topological relationships between features. In essence, a schematic network is a fully functional network without any bells or whistles.

5.1.11.3 Limitations

Currently, the tool builds a set of schematic lines, but does not build a network. That part is left to the user. In the future, geometric network creation may be included in a revision to the tool.

5.2 RETRIEVE NWIS DATA

This tool retrieves NWIS data from the Internet and builds time series tables from the data. The structure for the time series tables follows that of the TimeSeries class in the ArcGIS Hydro data model.

5.2.1 Description

The USGS National Water Information System (NWIS) contains historical daily streamflow information for USGS stream gages in the United States, with records dating back as far as 100 years. The data can be accessed from the Internet by inputting parameters such as gage number and period of record at the NWIS-Web site. The site then builds a URL and retrieves the data. The following table breaks down each component of a sample NWIS URL.

Sample URL:

http://waterdata.usgs.gov/nwis-w/TX/data.components/hist.cgi?statnum=08159000&bdate_month=03&bdate_day=23&bdate_year=1976&edate_month=09&edate_day=30&edate_year=1999&graphsize=1.5&mode=data&dateformat=0

Component	Meaning (Example)
http://waterdata.usgs.gov/nwis-w/	Base address of NWIS-Web site
TX	State that the gages are located in (TX)
statnum=08159000	Station number (08159000)
bdate_month=03	Start month (March)
bdate_day=23	Start day (23rd)
bdate_year=1976	Start year (1976)
edate_month=09	End month (September)
edate_day=30	End day (30th)
edate_year=1999	End year (1999)
graphsize=1.5&mode=data&dateformat=0	Output format (Tabular Data)

Table 5.3 Components of NWIS-Web URL

The data is retrieved based on the contents of the URL, as opposed to a hidden processing of the user's inputs. This means that if the user is familiar with the NWIS-Web URL format, and if the necessary parameters for data retrieval are known, then the user can type the URL in the web browser and jump directly to the data of interest without having to fill in each parameter at the NWIS-Web site. A more powerful application of this technology is the automation of NWIS-Web data retrieval. If a program can build the URL and possesses Internet capabilities,

then it can automatically retrieve NWIS data and process that data without the user ever having to open a web browser.

The Retrieve NWIS Data tool is an example of such an application. This tool reads USGS gage numbers from a set of gages in the GIS. It then prompts the user for a period of record to retrieve and the state that the gages are located in. With these inputs, the tool downloads the NWIS data from the Internet and builds a time series table using the structure of the TimeSeries class from the ArcGIS Hydro data model. During the Internet data transfer, a simple Splash screen appears.

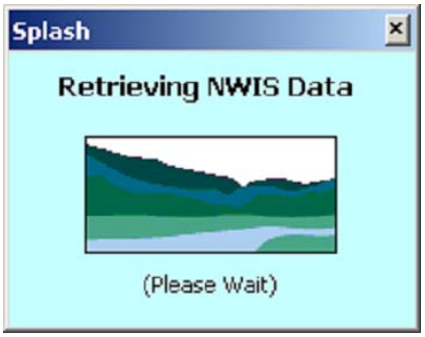

Figure 5.16 NWIS Splash Screen

If any of the gage numbers in the GIS do not correspond to valid USGS gage numbers, that number is skipped and the tool moves to the next gage. When the tool is finished, the time series table is added to the map document, and a message box appears giving some statistics about the tool's operation. If any data cannot be retrieved for any gages, the IDs of those gages are listed in a message box at the end of the tool's operation.

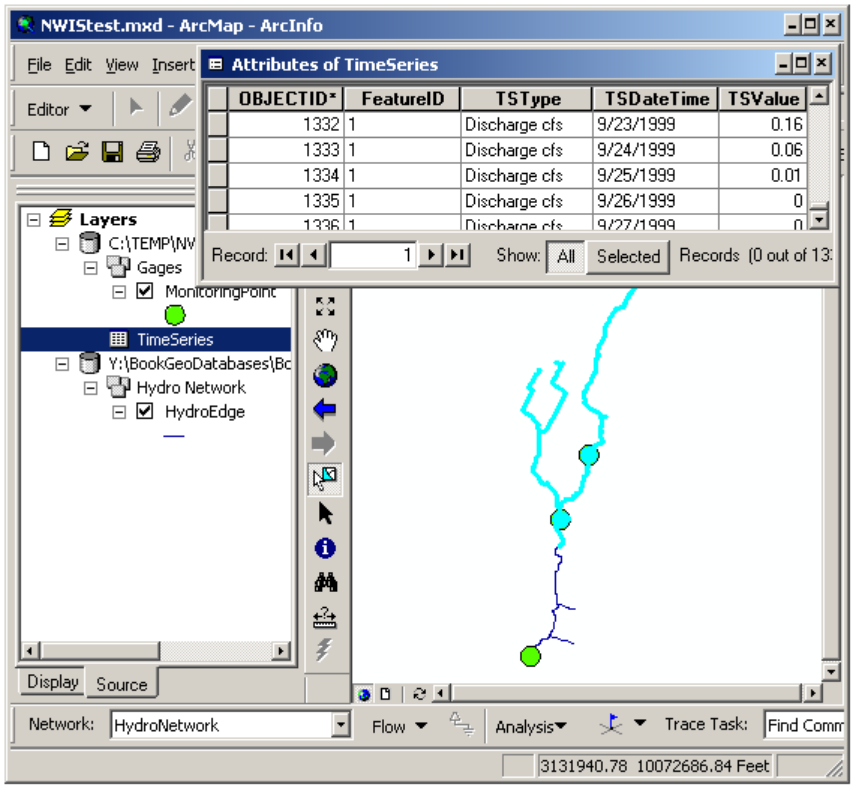

Figure 5.17 Table Resulting from Retrieve NWIS Data Operation

5.2.2 Beneficial Uses

This tool allows users to automatically retrieve time series information for USGS stream gages when needed. More importantly, this tool serves as a point of departure for the development of other tools that integrate GIS with automatic data retrieval from the Internet. One possible application would be a real-time flood forecasting model that would periodically retrieve time series information from the web, process the data, and alert the user if a flood hazard was predicted by the model. A second application relates directly to the drought that occurred in Texas in the summer of 2000. By accessing USGS real time data for streams,

reservoirs, and groundwater levels, a water-available model could be constructed that would assess the amount of water available in various sources throughout a given geographic region.

5.2.3 Limitations

The tool does not provide a mechanism for avoiding duplicate entries of time series data in the same table. This situation occurs if the user retrieved data for overlapping periods of record for the same gage station. The tool requires a connection to the Internet.

Chapter 6: Conclusions

The main contribution of this research is a prototype Arc Hydro toolset. This toolset is designed to operate with and populate the attributes of classes from the ArcGIS Hydro Data Model. However, the toolset is flexible enough to operate on any feature class, provided that it has the appropriate geometry and attribute structure.

To populate the attributes of Arc Hydro, each tool must not only operate within the functionality provided by the ArcGIS software system, but also follow the general principle or concept behind the attribute itself. For instance, length downstream may be interpreted in a number of ways, such as the average length of all downstream branches, or the length to the nearest sink. In the case of Arc Hydro, length downstream refers to the sum of the lengths of all downstream branches. While this method is relatively simple to implement, it will result in length downstream values that are greater than expected if there are downstream loops in the network. This tool could be improved by calculating length downstream as shortest path between the item of interest and the nearest sink. The shortest path is determined by reading weights from each network element that is traversed during the trace. The combination of network elements that connects the two end points while resulting in the lowest sum of weights is the shortest path. While this definition eliminates ambiguity as to what is really returned from a length downstream calculation, it requires the user to include an extra step (specifying the attribute to use as a weight) when creating the geometric

network, as well as increased processing time (to check path lengths to all connected sinks.) Other variations on the definition of length downstream could require even more effort from the user, or even more processing power and time from the computer. Thus when defining an attribute in Arc Hydro, care should be taken to insure that the attribute is meaningful to the water resources community, is simple for the user to populate, and can be produced unambiguously by the software's functionality.

Another concern for the developer is the logic behind the implementation of each tool. For example, a watershed outlet may be defined as the point where a watershed connects to the network. In many cases, this point is located by examining where the watershed and the network intersect. However, some cases may arise in which the watershed and the network intersect in more than one place, or even not at all. A tool designed to locate watershed outlets should either possess a scheme for dealing with these cases, or skip these cases during processing and provide a list of unprocessed watersheds as output. Because several schemes may exist to solve a given problem, the user should always be provided with information about how a tool operates, especially when it operates outside of normal conditions. The best practice is to clearly define how an attribute is interpreted with each possible situation from the start, and then leave it to the user to resolve unanticipated ambiguities.

These tools work very well with simple networks. However, the effectiveness of the tools may diminish with increasingly complex networks. Branches, loops, and Complex Edges may create ambiguities that demand more

stringent rules for trace task operations in order to produce the desired result. In most cases, these ambiguities can be resolved. A more serious problem involves operations on networks with more than one Edge Feature Class. If a tool utilizes an attribute or weight from network edges, then those attributes and weights must be defined and specified for all Edge Feature Classes in the network. Preparing for the possibility of multiple Edge Feature Classes in a network greatly increases development efforts. In addition, the graphical user interface becomes more complex, and the overall efficiency of the tool drops as each Edge Feature Class is processed for a given cycle of an operation.

Due to the complexity involved in trying to prepare the toolset for every possible situation, a better development approach is to design the tools to work strictly on Arc Hydro Feature Classes. Inherent in Arc Hydro's design is a set of rules, which provides a well-defined environment that each tool can expect to operate in. While the tools may allow flexibility in the names of attributes or Feature Classes, the network and other components to which the tools are applied should conform to the structure of Arc Hydro (such as one and only one Edge Feature Class in the network.)

In addition to improving the philosophy behind the design of the toolset (and Arc Hydro), the functionality of several of the tools may be improved as well. Currently, length downstream is calculated for an edge by summing the lengths of all edges produced in a downstream trace from that edge. This length would include both sides of a downstream loop, resulting in a value higher than what should be produced according to Arc Hydro's definition of length

downstream. The Calculate Length Downstream tool should be modified to find the least cost path based on edge length between a given edge and the nearest sink. It should also be modified to calculate length downstream for edges, or junctions, or both. Currently, length downstream must first be calculated for edges and then copied to junctions. This is not a very efficient design since it requires one tool to be run before the other. Because the process for calculating length downstream is identical for edges and junctions, the tool should allow length downstream to be calculated for both network element types.

The Assign Flow Direction tool could be improved by adding a separate button, which allows the user to change the flow direction of selected edges without having to open the graphical user interface for the tool. In other words, the user would choose the flow direction that will be assigned to edges. From that point forward, each time the button is pressed, that flow direction will be assigned to the selected edges.

Use of the NextDownstream attribute on HydroJunctions leads to an important concept: navigation of the landscape based on an attribute. This concept can be extended to include a NextDownstream attribute for a variety of Feature Classes. Thus, Watersheds could be connected to HydroJunctions, which could be connected to HydroEdges or a UserPoint. A tool could be written to navigate these Feature Classes based on the NextDownstream attribute. Such a tool would probably require a "HydroConnectivity" table, which contains the HydroIDs of Features and the HydroIDs of each feature's next downstream element. This concept could also lead to an improvement in the Make Schematic

Lines tool. This tool could be modified to create a set of points and lines which connect each feature involved in the HydroConnectivity table.

On a more general note, this research has proven that effective tools can be developed to work in ArcGIS by creating a DLL in the Visual Basic programming environment. While creating a water resources computational model which operates on top of the ArcGIS Hydro Data Model is certainly a more complex task, it is certainly within the realm of possibilities. The model developer must carefully consider to what extent the model is linked with ArcGIS, and how the graphic user interface is designed. In addition, the components of the model should be constructed in a COM-compliant and reusable fashion. Finally, care should be taken to separate the data from the computational components of the model. This approach helps to insure the integrity of the data and promotes a more modular design of the model.

6.1 FUTURE WORK

Each of the tools developed for this research should be considered a draft version. A good understanding of the proper protocols and techniques for developing tools in ArcGIS should be acquired directly from ESRI personnel, and then applied to the design of each tool to ensure the smoothest link with the ArcGIS environment.

The success of the tools in populating attributes of the ArcGIS Hydro Data Model suggests that the functionality of the tools may be incorporated into Data Model classes as methods. This process would involve generating code in Visual C++ using the Code Generation Wizard. Methods such as calculating length

downstream and storing flow direction are relatively straightforward and could be added to the HydroEdge class. However, methods such as accumulating area values to outlets or storing area outlets may require some thought as to where the methods should go and how they should be implemented.

Once some elementary methods have been written, the next step in the development of Arc Hydro is to use Arc Hydro to prepare data for a simulation model. Then a true evaluation of Arc Hydro's effectiveness can be ascertained.

Several innovations regarding GIS technology and the potential to link GIS data with computational models have been described. COM-compliance has made GIS the newest member of a family of COM-compliant software that can share components and utilize each other's object libraries. Object-oriented programming has extended the power of Features in GIS. HEC's libHydro has provided geoobjects with a great resource of efficient and well-established hydrologic computational routines. The next step is to take each of these elements and build the next generation of GIS-enabled hydrologic and hydraulic models.

Appendix A: ArcHydro and NWIS Tools User Guide

A.1 INSTALLING ARCHYDRO TOOLS

The ArcHydro Tools toolbar contains all of the ArcHydro Tools organized into toolbar items and menus. Once installed, the ArcHydro Tools toolbar can be added to an ArcMap document like any other ArcMap toolbar.

Installing ArcHydro Tools requires the following software:

ArcInfo 8.1 or higher

Installing ArcHydro Tools requires the following files:

ArcHydroTools.dll
ArcHydroTools.lib
ArcHydroTools.exp

These files, as well as all source files, can be found on the CD-ROM included with this thesis. Once the files have been extracted from CD-ROM, open the ArcMap document where ArcHydro Tools are to be installed.

- Click the Tools menu and then click Customize.
- In the customization window, click Add From File.
- Navigate to ArcHydroTools.dll, click on ArcHydroTools.dll, and then click Open. After a moment, a list of added items will be shown in a popup window. Click OK to close the window and proceed.
- In the customization window under the Toolbars tab, place a check next to ArcHydroTools. The ArcHydro Tools toolbar will appear on the map document. You may now close the customization window.

ArcHydro Tools can also be added to ArcMap by registering the ArcHydroTools.dll with ArcMap using RegCat.exe, and then adding the toolbar

to ArcMap like any other ArcMap toolbar. See ArcMap documentation for details.

A.2 ASSIGN HYDRO ID

This tool assigns HydroIDs to Feature Classes and tables in the current map that have a HydroID field of type String. To use this tool, add all Feature Classes and tables that you want to assign HydroID to, to the ArcMap document. In the editor toolbar, click Start Editing. In the ArcHydro Tools toolbar, click ArcHydro Tools, then click Assign HydroID.

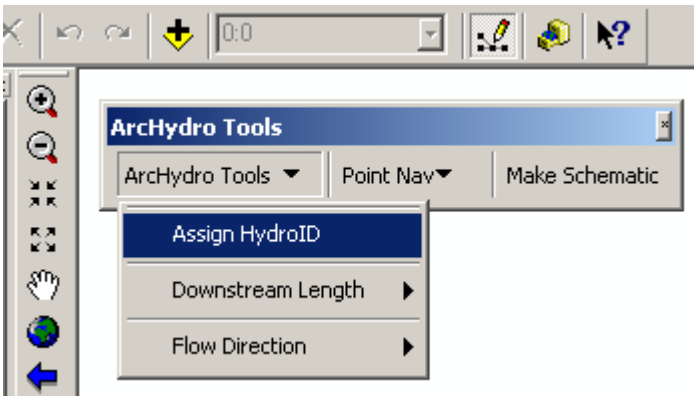

Figure A.1 Assign HydroID Button

The tool will work through each Feature Class and table in the map, assigning HydroIDs to records with a HydroID field of type String. You may also select features that you want to assign HydroID to. The tool will then only assign HydroIDs to selected features. A progress bar shows the tool's progress in the lower left corner of the map. When the tool is finished, investigate the tables to

ensure that HydroID was properly assigned. If you decide that you do not want HydroIDs, stop the edit session without saving edits.

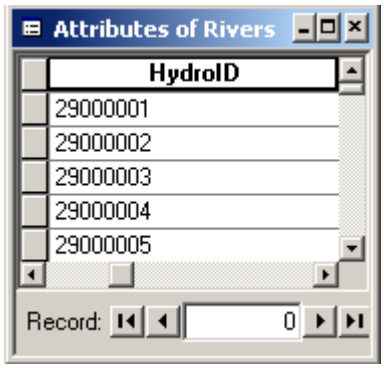

Figure A.2 HydroIDs for Rivers Feature Class

A.3 CALCULATE DOWNSTREAM LENGTH

This tool calculates the length from the most downstream node on each edge in a given layer to the sink that the edge flows to. It then populates the specified length downstream field with those calculated values. To use this tool, add the edge Feature Class to the map document. Start an edit session. Before using the Calculate Downstream Length tool, make sure that flow direction is set on the network. Once flow direction is set, on the ArcHydro Tools toolbar click ArcHydro Tools, then click Downstream Length, then click Calculate for Edges.

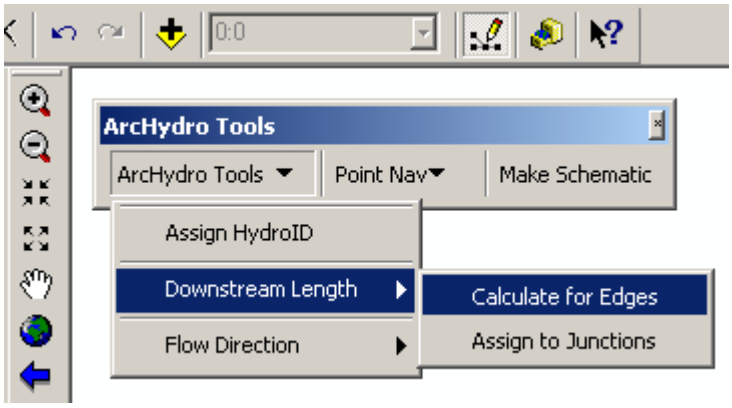

Figure A.3 Calculate Downstream Length Button

A form asking for the name of the edge layer, the field that contains the lengths of the edges, and the field to store downstream lengths appears.

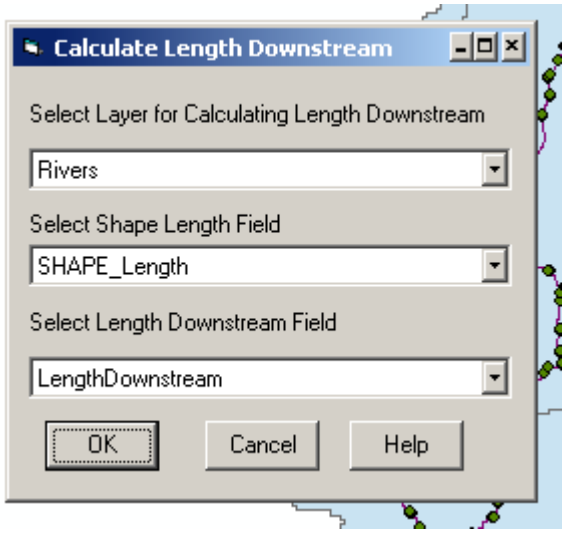

Figure A.4 Input Form for Calculate Downstream Length Tool

You may click the Help button to show information about the tool if you wish. Input the edge layer that you want to compute downstream lengths for in

the first box. Input the field in the edge layer that contains the lengths of each edge in the second box. These lengths are used to compute the downstream length for each edge. This field must be of type Double. Input the field in the edge layer that will hold the length downstream in the third box. This field must also be of type Double. Then click OK. The tool calculates values for a selected set of edges, or for the entire Feature Class if no edges are selected. A progress bar shows the tool's progress in the lower left corner of the map.

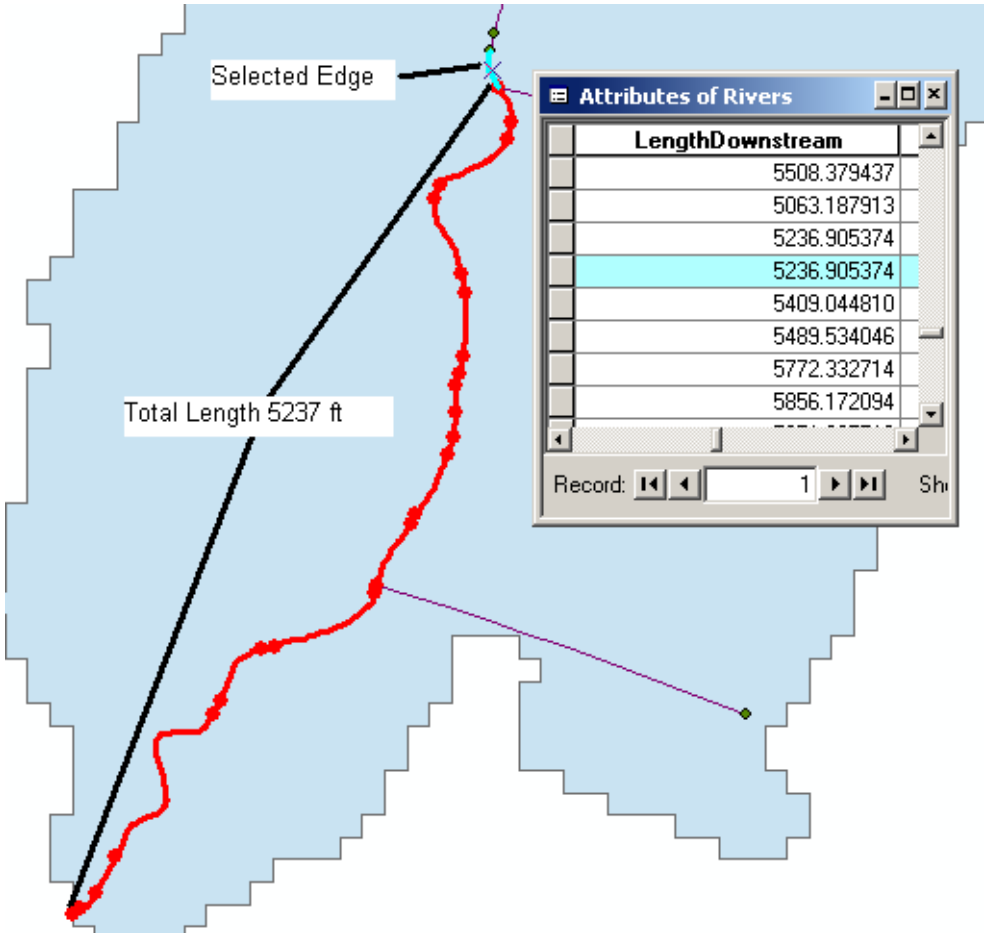

Figure A.5 Result of Downstream Length Computation

A.4 ASSIGN DOWNSTREAM LENGTH TO JUNCTIONS

The tool reads length downstream values from edges and writes those values to junctions in the network. To use this tool, add the network with the junction Feature Class of interest to the map document. Start an edit session. Make sure that downstream length has already been calculated for the edges in the network. On the ArcHydro Tools toolbar click ArcHydro Tools, then click Downstream Length, then click Assign to Junctions.

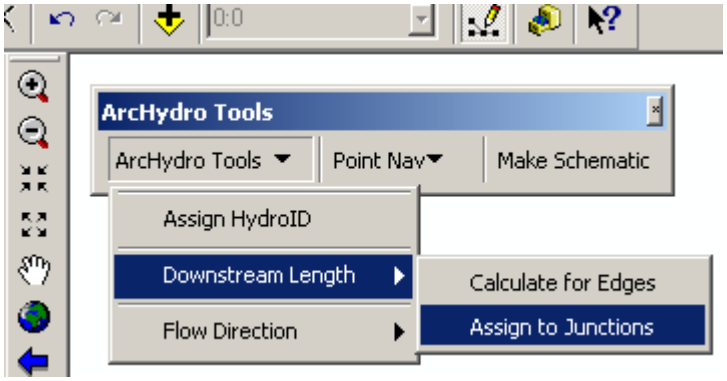

Figure A.6 Assign Downstream Length to Junctions Button

A form asking for layers and fields used by the tool appears.

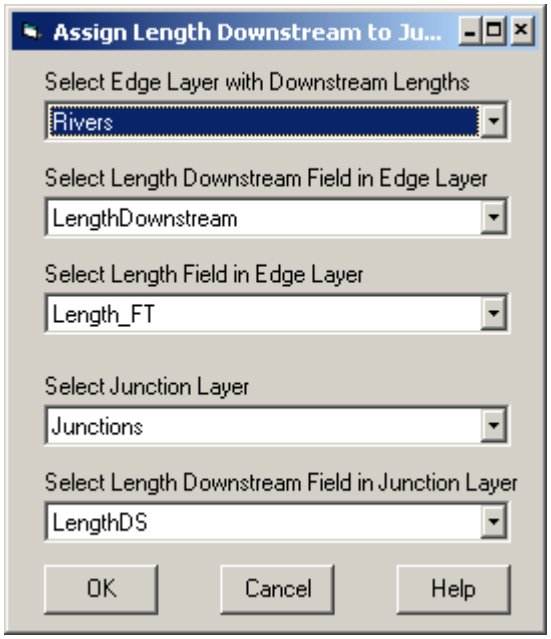

Figure A.7 Input Form for Copy Downstream Length to Junctions Tool

You may click the Help button to show information about the tool if you wish. Input the edge layer containing length downstream values in the first box. Enter the field in the edge layer where the length downstream values are stored in the second box. This field must be of type Double. Enter the shape length field for the edge layer in the third box. This field must be of type Double. Enter the junction layer where you want to write length downstream values to in the fourth box. Enter the field in the junction layer where you want to store length downstream values in the fifth box. This field must be of type Double. Then click OK. The tool calculates values for a selected set of junctions, or for the entire Feature Class if no junctions are selected. A progress bar shows the tool's progress in the lower left corner of the map.

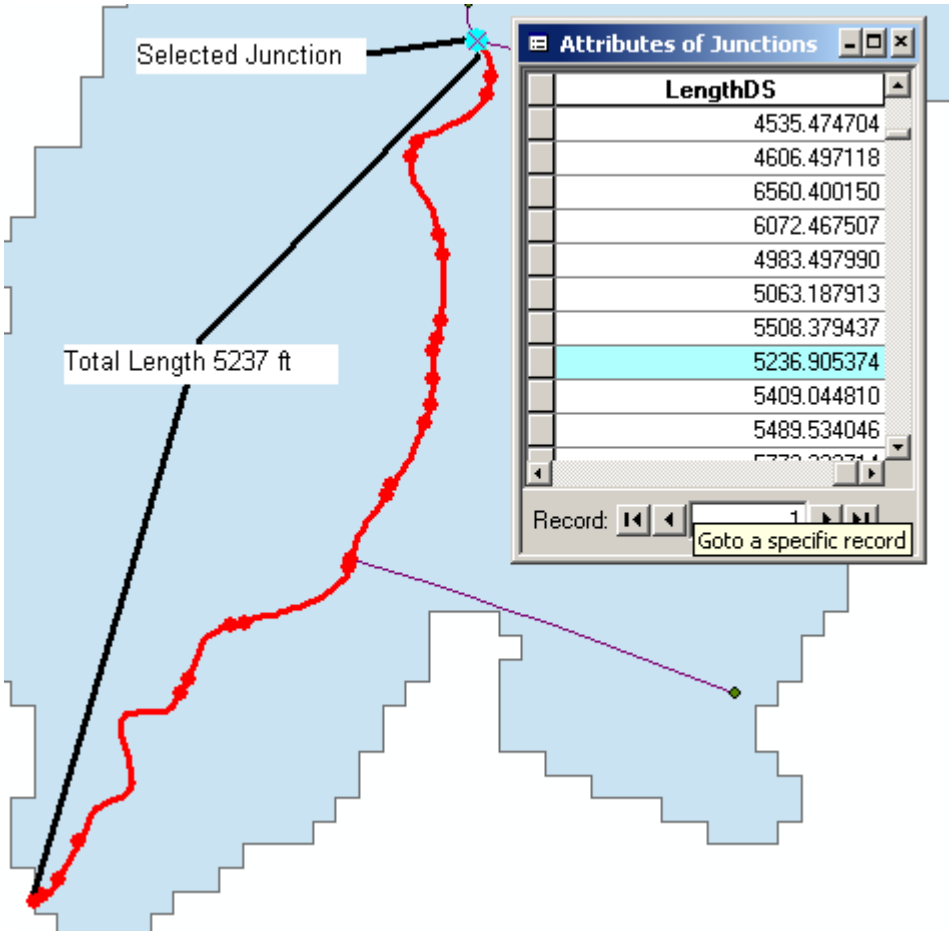

Figure A.8 Result of Assign Downstream Length to Junctions Computation

A.5 ASSIGN FLOW DIRECTION

This tool assigns flow direction to edges in a network, either by choosing a flow direction to assign, or by reading values from a table. To use this tool, add the network with the edge Feature Class of interest to the map document. Start an edit session. On the ArcHydro Tools toolbar click ArcHydro Tools, then click Flow Direction, then click Assign Flow Direction.

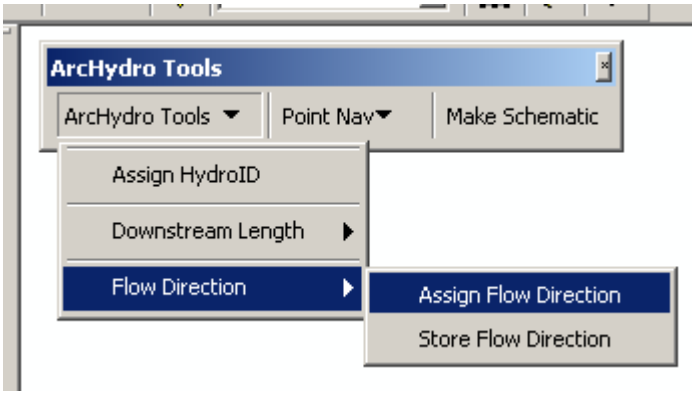

Figure A.9 Assign Flow Direction Button

A form asking for an edge layer and assignment options appears.

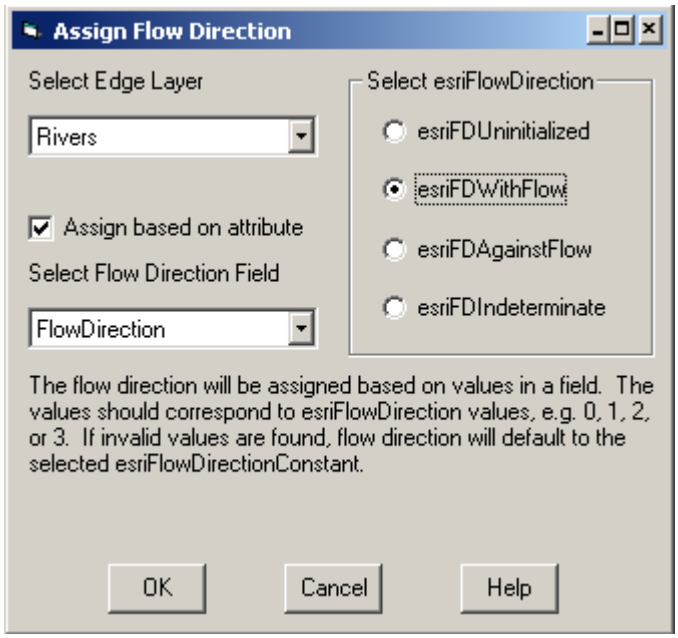

Figure A.10 Input Form for Assign Flow Direction Tool

You may click the Help button to show information about the tool if you wish. Input the edge layer that you want to assign flow direction to in the first

box. Select an esriFlowDirection from the options on the right. If not assigning flow direction based on an attribute, flow direction for the entire set of edges will be assigned based on the chosen esriFlowDirection. If assigning by attribute, place a check by the Assign based on attribute check box. The Select Flow Direction Field combo box will become enabled. The flow direction will be read from the field selected in the combo box. This field must be of a numeric type. With the desired options selected, click OK. The tool assigns flow direction to either a selected set of edges, or for the entire Feature Class if no edges are selected. A progress bar shows the tool's progress in the lower left corner of the map.

The figures below show how the Assign Flow Direction tool can be used to set the flow direction of edges in the network. The first figure shows the default flow direction based on sinks in the network. The second figure shows the flow moving towards a different location after the tool was used.

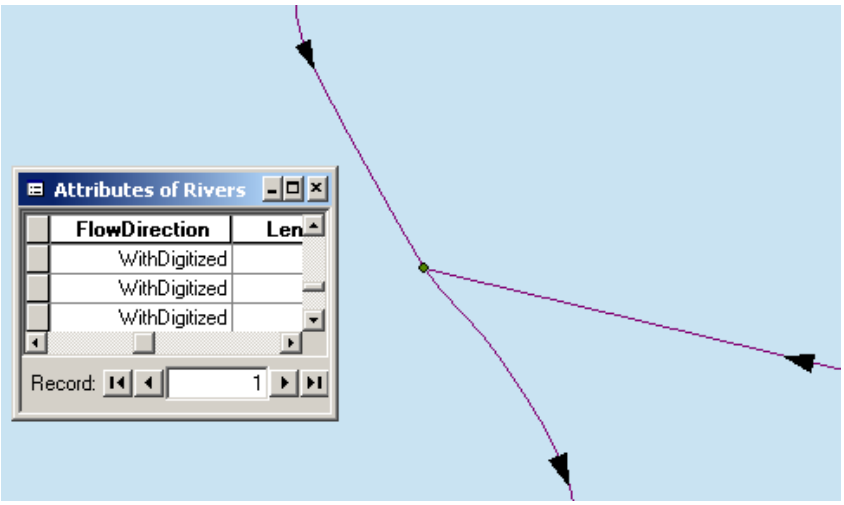

Figure A.11 Default Flow Direction in a Network

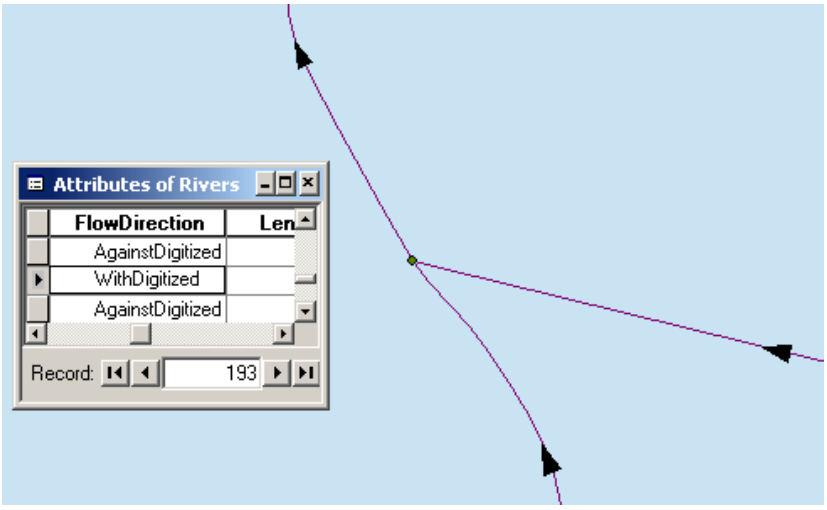

Figure A.12 Result of Changing Flow Direction with Assign Flow Direction Tool

A.6 STORE FLOW DIRECTION

This tool reads the flow direction for a set of edges from the network and writes the value of the flow direction to the Feature Class. To use this tool, add the network with the edge Feature Class of interest to the map document. Start an edit session. Make sure that flow direction has already been set for the network. On the ArcHydro Tools toolbar click ArcHydro Tools, then click Flow Direction, then click Store Flow Direction.

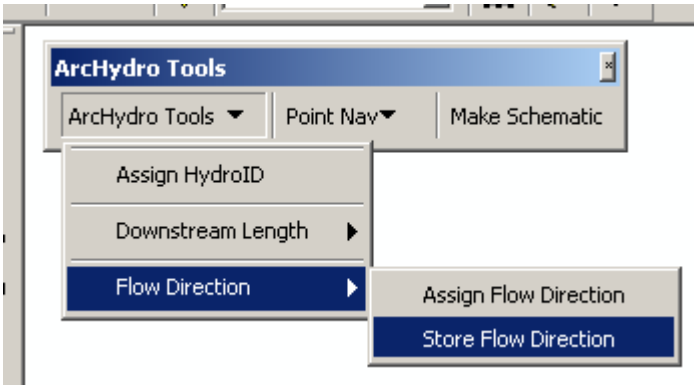

Figure A.13 Store Flow Direction Button

A form asking for an edge layer and a field to store flow direction appears.

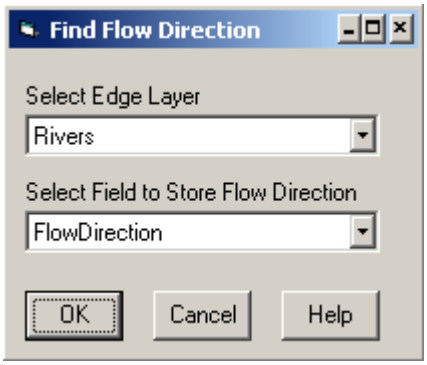

Figure A.14 Input Form for Store Flow Direction Tool

You may click the Help button to show information about the tool if you wish. Input the edge layer that you want to store flow direction for in the first box. Enter the field in the edge layer where the flow direction values are to be stored in the second box. This field must be of a numeric type. Then click OK. The tool reads flow directions values from the network and writes them to the Feature Class table. The tool finds values for a selected set of edges, or for the

entire Feature Class if no edges are selected. A progress bar shows the tool's progress in the lower left corner of the map. Results for storing flow direction for a sample set of edges are shown below. Note that a coded value domain has been applied to the FlowDirection field. Instead of displaying the numeric value stored in the field, a text description of the value's meaning is shown.

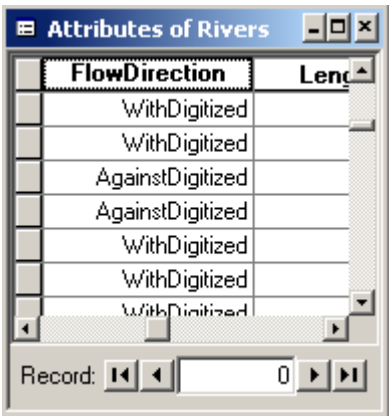

Figure A.15 Results of Store Flow Direction Operation

A.77 FIND NEXT DOWNSTREAM

The Find Next Downstream tool uses the network to find the next downstream junction in a particular Feature Class, and assigns the ID of the downstream junction to a downstream junction ID field in the Feature Class table. To use this tool, add the network with the junction Feature Class of interest to the map document. Start an edit session. On the ArcHydro Tools toolbar click Point Nav, then click Find Next Downstream.

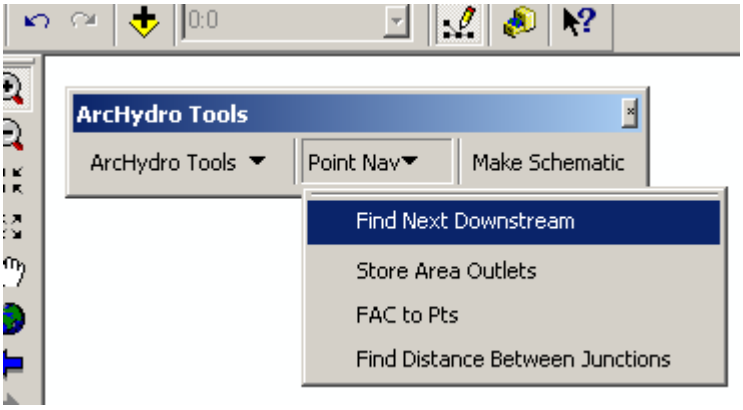

Figure A.16 Find Next Downstream Button

A form asking for a junction layer and required fields appears.

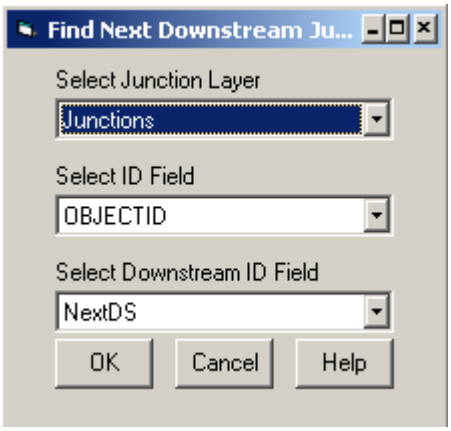

Figure A.17 Input Form for Find Next Downstream Tool

You may click the Help button to show information about the tool if you wish. Input the junction layer that you want to process in the first box. Enter the field that holds the junction IDs in the second box. Enter the field to store the downstream junction IDs in the third box. The field type of this field should match up with the field type of the ID field. Next click OK. The tool finds the

next downstream junctions and assigns the IDs of those junctions to the downstream junction ID field in the junction Feature Class table. The tool finds next downstream junctions for either a selected set of junctions, or for the entire Feature Class if no junctions are selected. A progress bar shows the tool's progress in the lower left corner of the map.

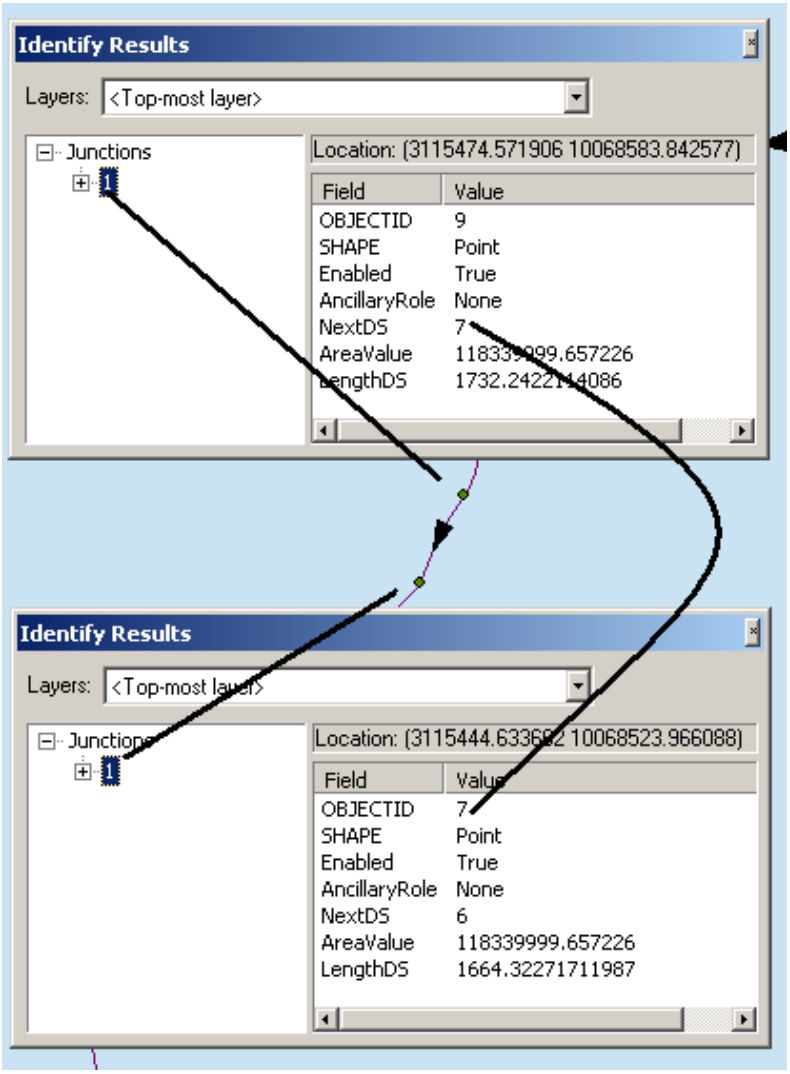

Figure A.18 Results of Find Next Downstream Computation

A.8 STORE AREA OUTLETS

This tool determines the outlet junction for an area and assigns the ID of that junction to an outlet ID field in the area Feature Class table. To use this tool, add the area Feature Class of interest and the network with the junction Feature Class of interest to the map document. Start an edit session. On the ArcHydro Tools toolbar click Point Nav, then click Store Area Outlets.

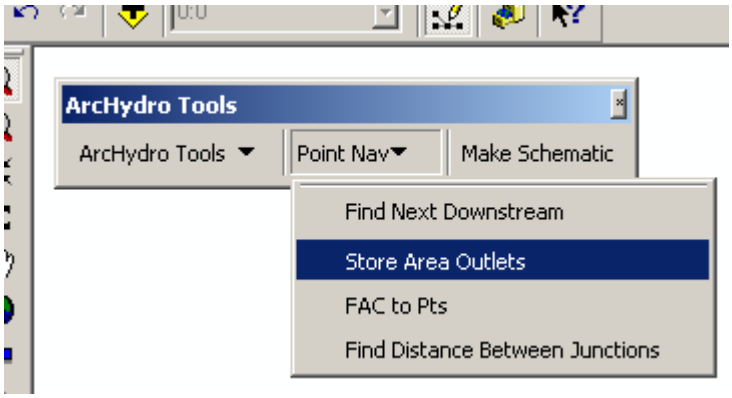

Figure A.19 Store Area Outlets Button

A form asking for the required layers and fields appears.

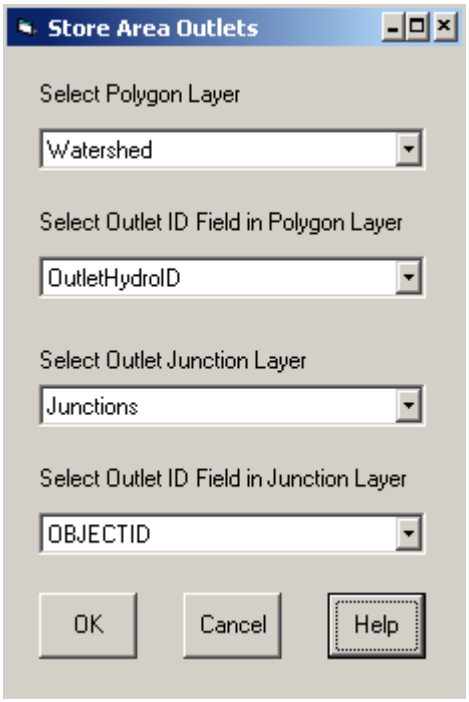

Figure A.20 Input Form for Store Area Outlets Tool

You may click the Help button to show information about the tool if you wish. Input the polygon layer that represents the areas of interest in the first box. Input the field to store outlet IDs in the area Feature Class in the second box. Input the junction layer that will serve as outlets for the areas in the third box. Input the field in the junction Feature Class that contains the IDs that will be written to the Outlet ID field in the area Feature Class in the fourth box. Then click OK. The tool uses the scheme described above to assign outlet IDs to the areas of interest. The tool runs on either a selected set of areas, or for the entire Feature Class if no areas are selected.

During processing, a form with a progress bar is displayed in the center of the screen. The user may press the Cancel button on the form to cancel

processing. When the tool is finished, a message box appears displaying the ObjectIDs of any areas for which the outlet could not be determined, or for which there was more than one potential outlet. If no such areas were found, then the message box does not appear.

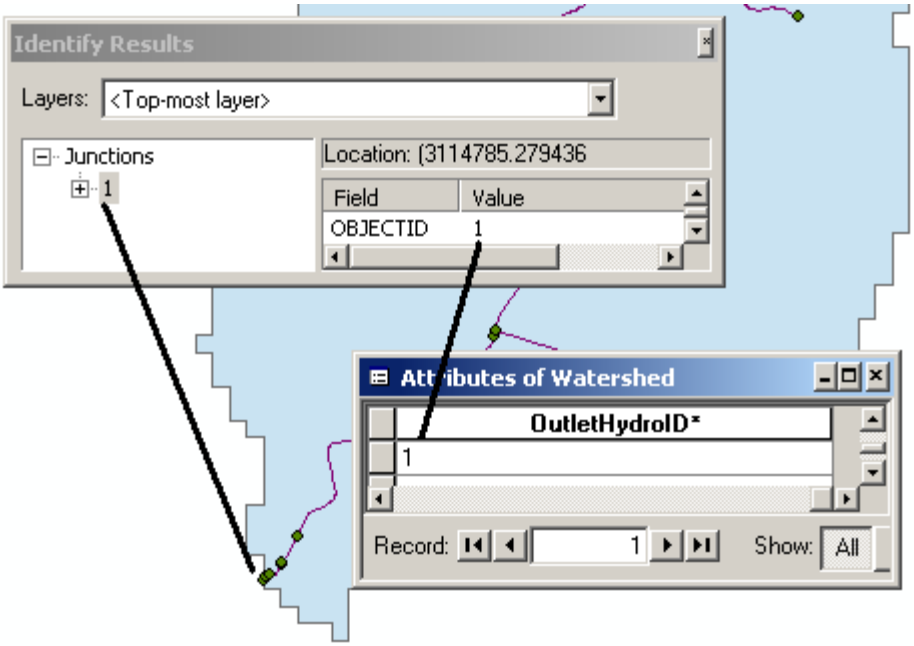

Figure A.21 Results of Store Area Outlets Computation

A.9 ACCUMULATE AREAS TO POINTS

This tool accumulates values from areas to the outlet junction of the areas, and then accumulates those values through each downstream junction in the network. To use this tool, add the area Feature Class of interest and the point or junction Feature Class of interest to the map document. Start an edit session. On the ArcHydro Tools toolbar click Point Nav, then click FAC to Pts.

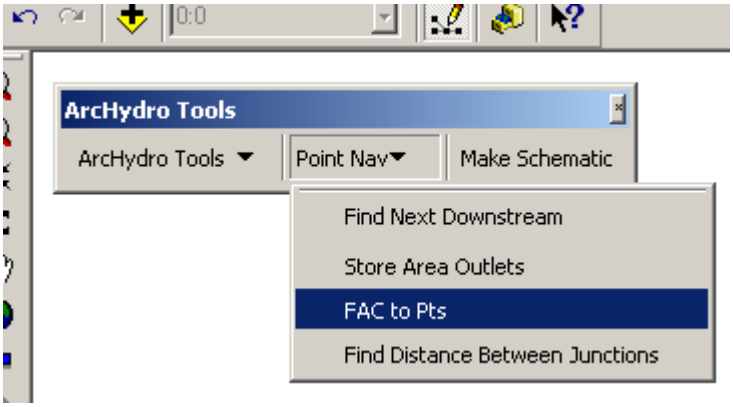

Figure A.22 Accumulate Areas to Points Button

A form asking for the required layers and fields appears.

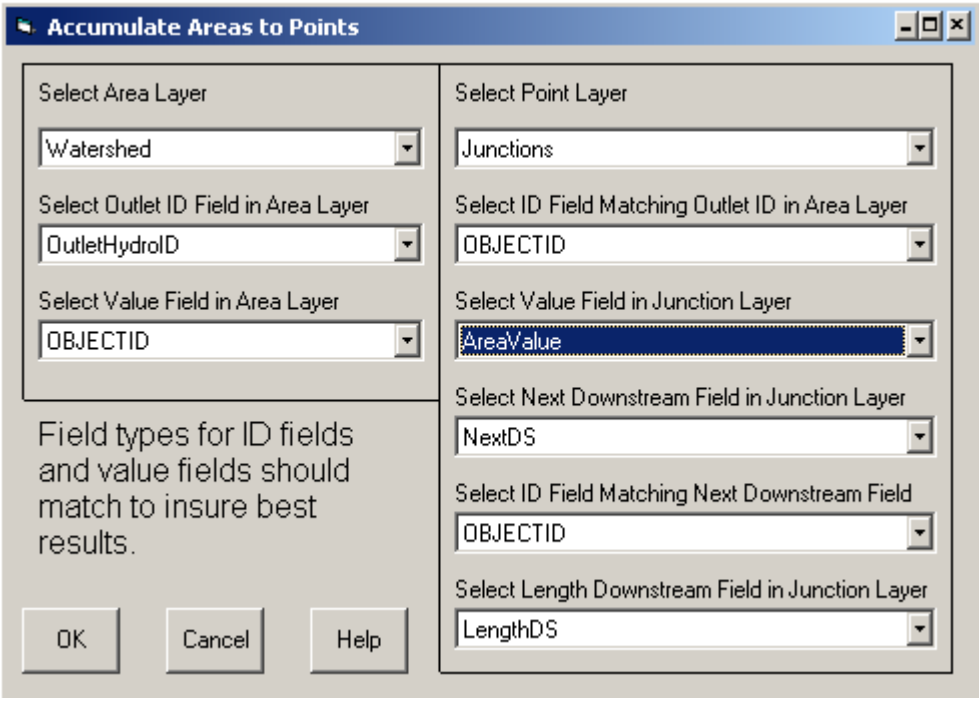

Figure A.23 Input Form for Accumulate Areas to Points Tool

You may click the Help button to show information about the tool if you wish. Input the polygon layer that represents the areas of interest in the first box in the column on the left. Input the field that contains outlet IDs in the area Feature Class in the second box. Input the field that contains the values to be summed in the third box. In the column on the right, input the junction layer that serves outlets for the areas in the first box. Input the field in the junction Feature Class that contains the IDs that correspond to the Outlet ID field in the area Feature Class in the second box. Input the field that will store the values from the areas in the third box. Input the field that contains the Next Downstream IDs in the fourth box. Input the field that corresponds to the IDs in the Next Downstream IDs field in the fifth box. Input the field that contains length downstream in the sixth box. The ID fields should be of type Integer or String, and they should match in type. The Length Downstream field must be of type Double. The field that stores the values from the areas may be numeric, String, or Date. With all fields entered, click OK.

The tool uses the scheme described above to accumulate values from areas onto points, and then passes those values through the network in the downstream direction. The tool operates on the entire set of points or junctions. During processing, a form with a progress bar is displayed in the center of the screen. The user may press the Cancel button on the form to cancel processing.

The following figure shows the results of running the tool where the accumulated value was the area of each polygon in acres. The figure shows the outlet point for the most upstream area in the network.

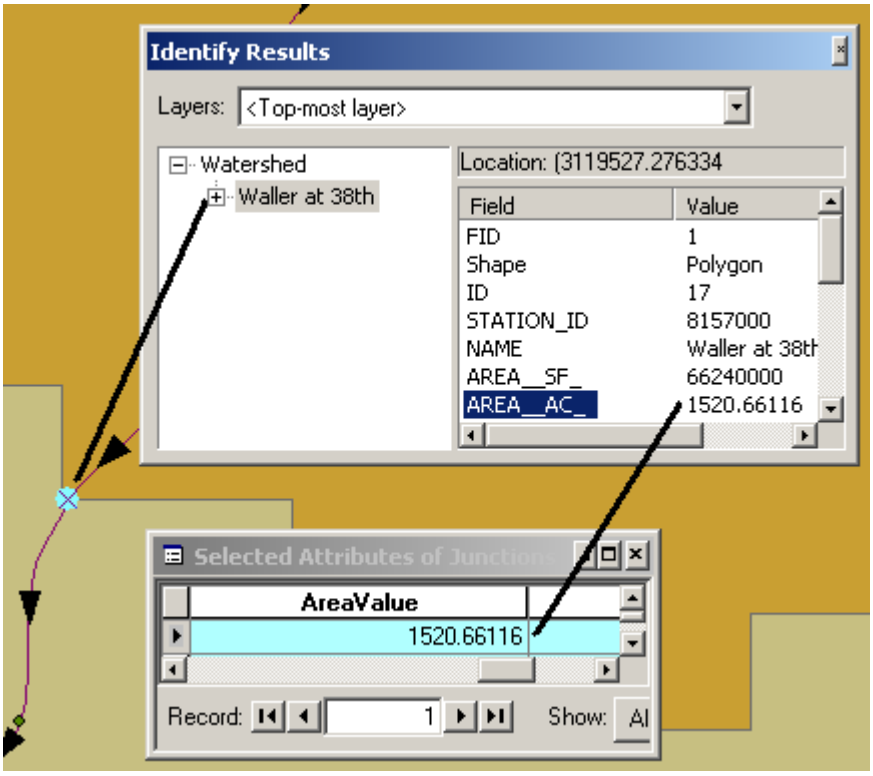

Figure A.24 Results of Accumulate Areas to Points Computation

A.10 FIND DISTANCE BETWEEN JUNCTIONS

This tool finds the distance between a point or junction and its next downstream neighbor. To use this tool, add the point or junction Feature Class of interest to the map document. Start an edit session. On the ArcHydro Tools toolbar click Point Nav, then click Find Distance Between Junctions.

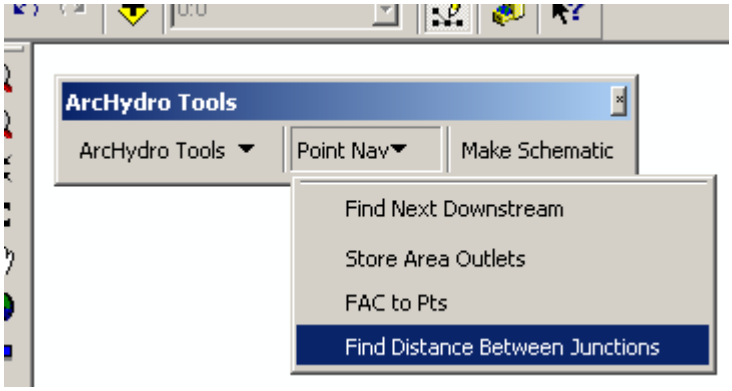

Figure A.25 Find Distance Between Junctions Button

A form asking for the required layer and fields appears.

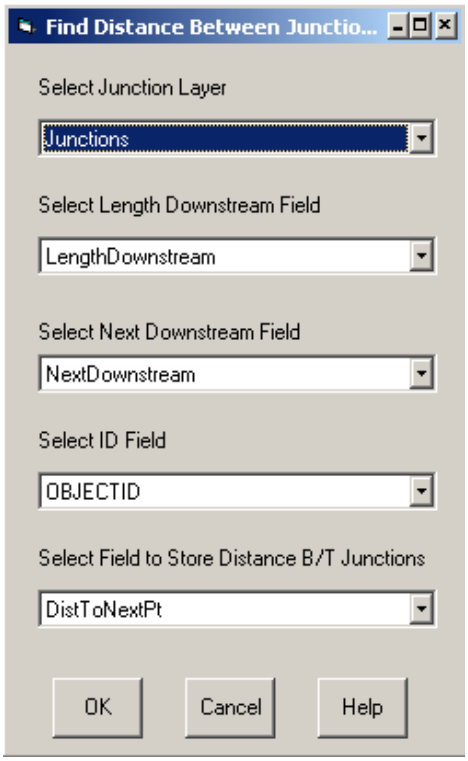

Figure A.26 Input Form for Find Distance Between Junctions Tool

You may click the Help button to show information about the tool if you wish. Input the junction or point layer in the first box. Input the field that contains length downstream in the second box. This field must by of type Single or Double. Input the field that contains the next downstream IDs in the third box. Input the field that contains the IDs that correspond to the next downstream IDs in the fourth box. The ID fields must be of type String or Integer, and they should match. Input the field to store the distance between a junction and its downstream junction in the fifth box. This field must be of type Single or Double. With all parameters inputted, click OK.

The tool uses the scheme described above to find the distance between junctions. The tool operates on a selected set of points or junctions, or the entire set if none are selected. During processing, a form with a progress bar is displayed in the center of the screen. The user may press the Cancel button on the form to cancel processing.

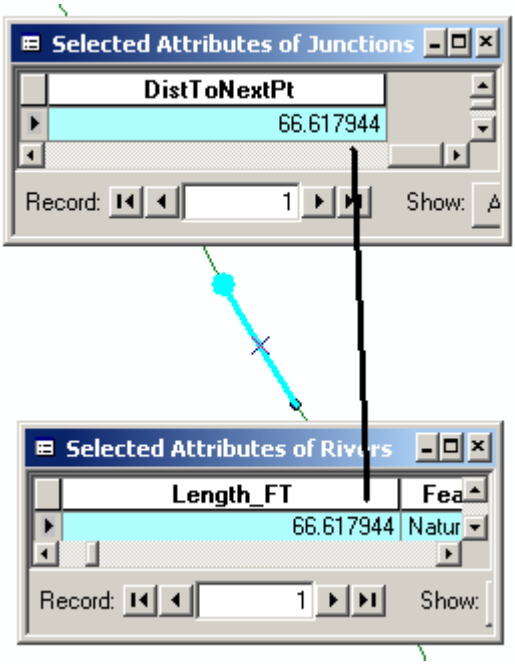

Figure A.27 Results of Find Distance Between Junctions Computation

A.11 MAKE SCHEMATIC LINES

This tool creates a schematic line Feature Class from a set of points or junctions. To use this tool, add the point or junction Feature Class of interest to the map document. Start an edit session. On the ArcHydro Tools toolbar click Make Schematic.

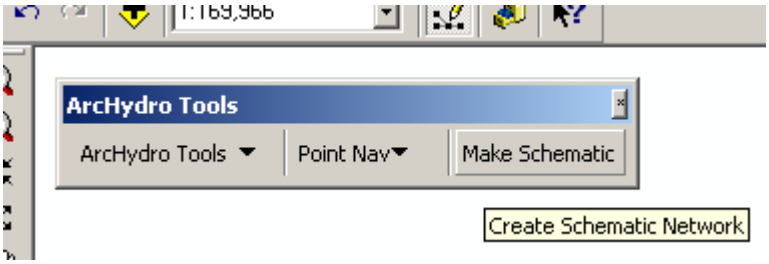

Figure A.28 Make Schematic Lines Button

A form asking for the required layer and fields appears.

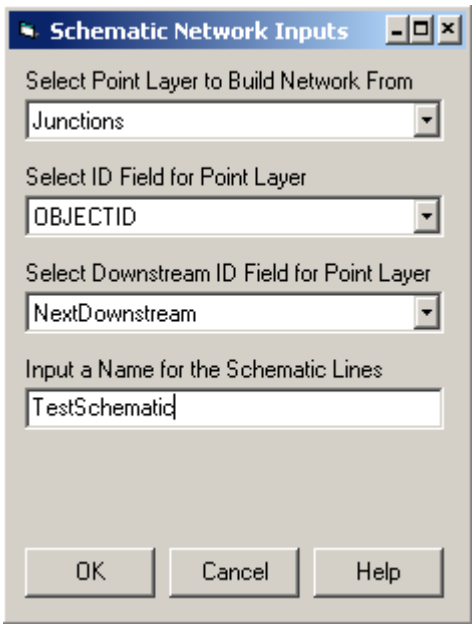

Figure A.29 Input Form for Make Schematic Lines Tool

You may click the Help button to show information about the tool if you wish. Input the junction or point layer in the first box. Input the field that contains the next downstream IDs in the second box. Input the field that contains the IDs that correspond to the next downstream IDs in the third box. The ID fields must be of type String or Integer, and they should match. Input a name for the schematic line Feature Class in the fourth box. Remember that this name should conform to a valid class name according to ESRI rules. Invalid class names will produce an error. With all parameters inputted, click OK.

The tool uses the next downstream ID to build a set of schematic lines between all points in the given point or junction Feature Class. A progress bar shows the tool's progress in the lower left corner of the map.

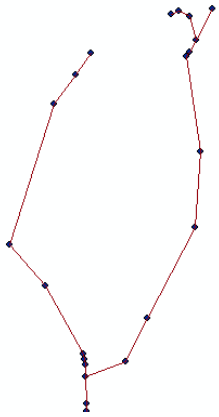

Figure A.30 Results of Make Schematic Lines Operation

A.12 NWIS TOOL

This tool retrieves NWIS data from the Internet and builds time series tables from the data. Once installed, the NWIS button can be added to an ArcMap document like any other ArcMap command button.

Installing the NWIS tool requires the following software:

ArcInfo 8.1 or higher

Installing the NWIS tool requires the following files:

NWIS.dll
NWIS.lib
NWIS.exp

These files, as well as all source files, can be found on the CD-ROM included with this thesis. Once the files have been extracted to the computer, open the ArcMap document where the NWIS tool is to be installed.

- Click the Tools menu and then click Customize.
- In the customization window, click the Commands tab.

  Note:

  In the lower left corner of the window, you will see a Save in… dropdown box. Make sure you are not saving in normal.mxt by clicking in the dropdown box and selecting the name of the map document. If you save in normal.mxt, then the NWIS tool will load for each new document that you create with ArcMap. This creates some extra overhead, especially if you do not need the NWIS tool in every new ArcMap document.

- Click on Get NWIS Data with the left mouse button.
- While holding down the left mouse button, drag the Get NWIS Data button next to another button in the gray area where all the commands and tools are in ArcMap. When you see the insertion cursor (looks like a capital I,) that means that tool can be placed in that location. Release the left mouse button to drop the tool in that location. The Get NWIS Data button is now ready for use.

To use this tool, add the Feature Class that contains the USGS stream gages of interest to the map document. The Feature Class table should contain a field that stores USGS stream gage ID numbers. Click the Get NWIS Data button.

Figure A.31 Retrieve NWIS Data Button

A form asking for a layer that represents USGS stream gages appears.

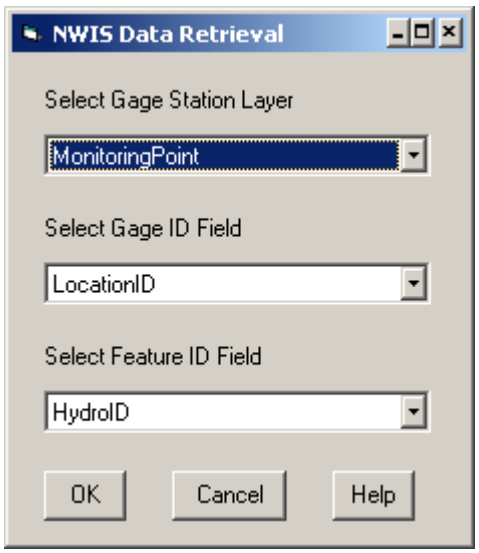

Figure A.32 Input Form for Layer and Fields

Input the Feature Layer that represents USGS stream gages in the first combo box. Input the field in that layer that stores USGS stream gage IDs in the second combo box. An example of a USGS stream gage ID is 08158000, which is for the Colorado River in Austin. Input the field that stores Feature IDs in the third combo box. Values in this field will be stored in the FeatureID field in the time series table, in order to link the time series data with the appropriate feature. Click OK. A form asking for the period of record to retrieve and the state the gages are located in appears.

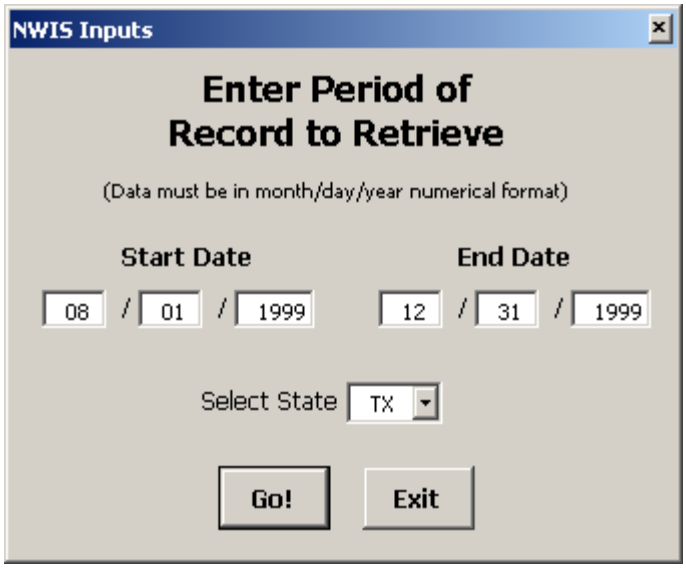

Figure A.33 Input Form for Period of Record and State

With the period of record and state inputted, press Go. The tool checks to make sure the dates exist, and that the start date occurs before the end date. The tool then asks for a database and table to store the data in. With these inputs, the tool then retrieves data from the Internet and builds the time series table. During the Internet transfer, the tool displays a simple Splash screen. During processing, a form with a progress bar is displayed in the center of the screen. The user may press the Cancel button on the form to cancel processing.

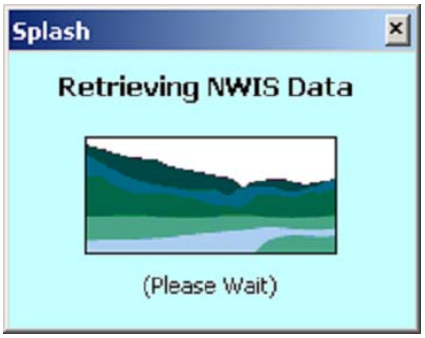

Figure A.34 NWIS Splash Screen

When the tool is finished, the table is added to the ArcMap document, and a message box appears giving some statistics about the table creation process. If data for any gages could not be downloaded, the IDs of those gages appear in a message box.

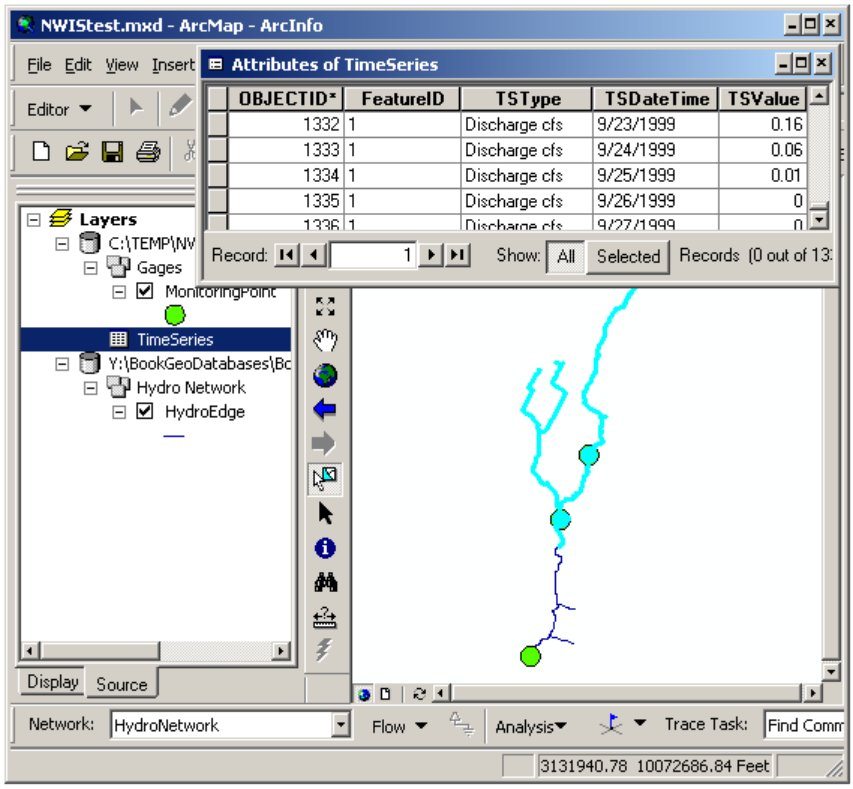

Figure A.35 Table Resulting from Retrieve NWIS Data Operation

Appendix B: Data Dictionary

Following is a list of files included on the CD-ROM attached to this thesis.

B.1 ARCHYDRO TOOLS

ArcHydroTools.dll	DLL containing ArcHydro Tools
ArcHydroTools.exp	Query expression for ArcHydroTools.dll
ArcHydroTools.lib	Lib file for ArcHydroTools.dll
ArcHydroTools.vbp	ArcHydro Tools VB Project
ArcHydroTools.vbw	ArcHydro Tools VB Workspace
clsArcHydroMenu.cls	ArcHydro Menu
clsArcHydroToolbar.cls	ArcHydro Toolbar
clsAsgFlowDir.cls	Assign Flow Direction Tool
clsCalcDSLength.cls	Calculate Downstream Length for Edges Tool
clsCopyDSLength.cls	Assign Downstream Length to Junctions Tool
clsFACtoPts.cls	Accumulate Area Values to Points Tool
clsFindDistBTPts.cls	Find Distance Between Points Tool
clsFindFlowDir.cls	Store Flow Direction Tool
clsFindNextDS.cls	Find Next Downstream Tool
clsMakeSchLines.cls	Make Schematic Lines Tool
clsMenuDSLength.cls	Downstream Length Menu
clsMenuFlowDir.cls	Flow Direction Menu

clsMxHydroID.cls	Assign Hydro ID Tool
clsPointNavMenu.cls	Point Nav Menu
clsStoreAreaOutlets.cls	Store Area Outlets Tool
frm2by1.frm	Template input form
frmAsgFlowDir.frm	Assign Flow Direction Input Form
frmBitmaps.frm	Bitmaps form
frmCalcDSL.frm	Calculate Downstream Length for Edges Input Form
frmCopyDSL.frm	Copy Downstream Length to Junctions Input Form
frmFACtoPts.frm	Accumulate Area Values to Points Input Form
frmFindDistBTPts.frm	Find Distance Between Points Input Form
frmFindFlowDir.frm	Store Flow Direction Input Form
frmFindNextDS.frm	Find Next Downstream Input Form
frmHelpASGFlowDir.frm	Assign Flow Direction Help Form
frmHelpCalcDSL.frm	Calculate Downstream Length for Edges Help Form
frmHelpCopyDSL.frm	Copy Downstream Length to Junctions Help Form
frmHelpFACtoPts.frm	Accumulate Area Values to Points Help Form
frmHelpFindDistBTPts.frm	Find Distance Between Points Help Form

frmHelpFindFlowDir.frm	Store Flow Direction Help Form
frmHelpFindNextDS.frm	Find Next Downstream Help Form
frmHelpStoreAreaOutlet.frm	Store Area Outlets Help Form
frmProgress.frm	Progress Form
frmSchematic.frm	Make Schematic Lines Input Form
frmSchematicHelp.frm	Make Schematic Lines Help Form
frmStoreAreaOutlet.frm	Store Area Outlets Input Form
modGlobals.bas	Global Variables Module
modNetworkUtils.bas	Network Utilities Module
modUtilsGeneral.bas	General Utilities Module
readmeArcHydroTools.txt	Basic readme file for ArcHydro Tools

B.2 NWIS TOOL

clsNWIS.cls	NWIS Tool
frmHelpNWISFields.frm	Fields Help Form
frmHelpNWISLayers.frm	Layers Help Form
frmNWISFields.frm	Fields Input Form
frmNWISinputs.frm	General NWIS Inputs Form
frmNWISLayers.frm	Layers Input Form
frmProgress.frm	Progress Form
frmSplash.frm	Splash Form
modArcHydro.bas	Module with other ArcHydro utilities
modGlobals.bas	Global Variables Module
modUSGSData.bas	USGS Data-sorting Module

modUtilsGeneral.bas	General Utilities Module
NWIS.dll	DLL containing NWIS Tool
NWIS.exp	Query expression for NWIS.dll
NWIS.lib	Lib file for NWIS.dll
NWIS.vbp	NWIS Tool VB Project
NWIS.vbw	NWIS Tool VB Workspace
readmeNWIS.txt	Basic readme file for NWIS Tool

B.3 SAMPLE TOOL

clsSampleTool.cls	Sample Tool
Project1.vbp	Sample Tool VB Project
Project1.vbw	Sample Tool VB Workspace
Sample_Tool.dll	Sample Tool DLL
Sample_Tool.exp	Query expression for Sample_Tool.dll
Sample_Tool.lib	Lib file for Sample_Tool.dll

Bibliography

Alfredsen, Knut, and Bjorn Saether. "An Object-oriented Application Framework for Building Water Resource Information and Planning Tools Applied to the Design of a Flood Analysis System." Environmental Modelling and Software. 15.3 (Mar. 2000): 215-224. 3 Mar. 2001 <http://www.sciencedirect.com/>.

Booch, Grady, James Rumbaugh, and Ivar Jacobson. The Unified Modeling Language User Guide. Reading: Addison-Wesley Longman, Inc., 1999.

Davis, Kim. Object-Oriented Modeling of Rivers and Watersheds in Geographic Information Systems. Thesis. The University of Texas at Austin, 1999. < http://www.ce.utexas.edu/prof/maidment/grad/davis/home.htm>

Donnelly, Kevin. Developing Digital Flood Insurance Rate Maps for Lago Vista. Thesis. The University of Texas at Austin, 2001.

ESRI. Arc Facilities Manager (ArcFM): A Powerful New ARC/INFO-Based Application for Utilities. ESRI White Paper. Environmental Systems Research Institute, Inc, 1998.

ESRI. ArcFM GIS for Utilities. CD-ROM. Environmental Systems Research Institute, Inc, 1999.

ESRI. ArcFM Water: AM/FM/GIS for Water Utility Systems. ESRI White Paper. Environmental Systems Research Institute, Inc, 2000.

ESRI. Building Applications with MapObjects. USA: Environmental Systems Research Institute, Inc., 1996.

Goodchild, M.F., and K.K. Kemp, eds. NCGIA Core Curriculum in GIS. National Center for Geographic Information and Analysis, University of California, Santa Barbara CA. 1990 <http://www.geog.ubc.ca/courses/klink/gis.notes/ncgia/u23.html#UNIT23 >.

Hartley, Tim. GUI Design Fundamentals Learning Guide. Phoenix: ComputerPREP, 1998.

HEC. HEC-GeoRAS: An Application for Support of HEC-RAS Using Arc/Info. Davis: Hydraulic Engineering Center, 1999.

HEC. HEC-HMS. Hydrologic Modeling System, Version 2.1. 17 Apr. 2001 <http://www.hec.usace.army.mil/software/software_distrib/hec-hms/hechmsprogram.html>.

HEC. libHydro Users Manual. Davis: Hydraulic Engineering Center, 1995.

Meyer, Bertrand. Object-Oriented Software Construction. Hertfordshire: Prentice Hall, 1997.

Murray, M.G., and V. Kutija. "Experiences in Object Oriented Design." Proceedings of 4th International Conference Hydroinformatics 2000. Cedar Rapids, Iowa. CD-ROM. 2000.

Olivera, Francisco. CRWR-PrePro: An ArcView Preprocessor for Hydrologic, Hydraulic and Environmental Modeling. 27 Apr. 2001 <http://www.ce.utexas.edu/prof/olivera/prepro/prepro.htm>.

Rumbaugh, James, et al. Object-Oriented Modeling and Design. Englewood Cliffs: Prentice Hall, 1991.

Spanou, Maria, and Daoyi Chen. "An Object-oriented Tool for the Control of Point-source pollution in river systems." Environmental Modelling and Software. 15.1 (Jan. 2000): 35-54. 3 Mar. 2001 <http://www.sciencedirect.com/>.

U.S. Army Corps of Engineers. National Inventory of Dams. 27 Apr. 2001 <http://crunch.tec.army.mil/nid/webpages/nid.cfm>.

Ye, Zichuan. Map-based Surface and Subsurface Flow Simulation Models: An Object-Oriented and GIS Approach. Diss. The University of Texas at Austin, 1996.

Vita

Timothy Lee Whiteaker was born in Cookeville, Tennessee on April 3, 1976, the son of Leigh Whiteaker and Charles Ed Whiteaker. After completing his work at Cumberland County High School, Crossville, Tennessee, in 1994, he entered Tennessee Technological University in Cookeville, Tennessee. He received the degree of Bachelor of Science in Civil and Environmental Engineering from Tennessee Technological University in May, 1999. In September, 1999, he entered The Graduate School at The University of Texas at Austin.

Permanent address: 523 Madison Ave.
Sparta, TN 38583

This thesis was typed by the author.